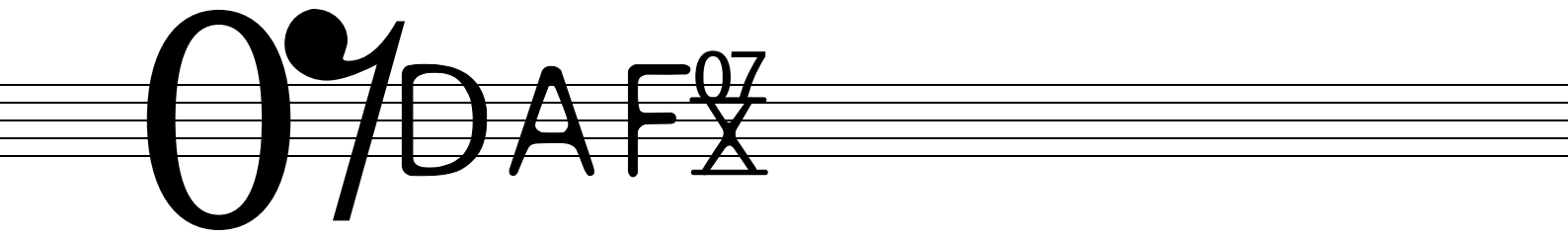
Proceedings of the 10th International Conference on Digital Audio Effects

DAFx-07



Credits
Cover Photos: André Raspaud
Cover Design: Bernadette de Tauzia and Martin Raspaud

ISBN: 978-88-901479-1-3

Copies may be ordered from:
Sylvain Marchand
LaBRI, Université Bordeaux 1
351, cours de la Libération
33405 Talence cedex
France
E-mail: dafx07@labri.fr

DAFx-07 homepage: `http://www.dafx.u-bordeaux.fr/`

Welcome to DAFx-07!

On behalf of all the members of the organizing committee, I am pleased to welcome you to the 2007 International Conference on Digital Audio Effects (DAFx-07). The DAFx conference originally started as a part of a very successful European COST G6 action, and is now standing – even running – on its own two feet as a self-funded event.

This is the tenth edition of this conference, each year being a new opportunity to maintain the high level of scientific excellence together with an open and friendly atmosphere. Although the conference is mainly concerned with digital audio effects, in fact it covers nearly all topics of digital audio and music processing.

As in the previous editions, the conference schedule has been designed to keep paper presentations in non-parallel sessions, with extended time periods reserved for poster sessions in the afternoon. This year, forty-eight works have been selected for presentation – thirty-seven papers and eleven posters. Many thanks go to the researchers who submitted their papers, to the DAFx scientific committee, and to the reviewers for their hard work.

This year's conference schedule will be complemented by two keynote talks given by leading specialists:

- Jean-Marc Jot, from Creative Labs / Creative Advanced Technology Center, California, USA;
- Xavier Serra, from Music Technology Group, Pompeu Fabra University, Barcelona, Spain.

I would like to thank the DAFx community for faithfully coming back each year to make this conference a major scientific event. I hope that new delegates will enjoy the conference and join the community. Regarding the open and friendly atmosphere, the social events are traditionally included in the registration. Thus, the concert and the banquet are always great opportunities for people to meet, artists, scientists, students, and leading composers and researchers, all in the same place.

Believe me, the DAFx conference really has something special! Xavier Serra was the Conference Chair of the first DAFx edition in Barcelona, where I gave my very first talk, as a PhD student. Then I grew up scientifically with DAFx, attending other famous conferences but without forgetting to present a paper at each edition of DAFx. In the meantime, I became an associate professor and got more and more involved in the DAFx community. DAFx has played a key role in my professional life. Now, I feel it is my turn to give a part of my time to contribute to the organization of this great conference.

Finally, I would like to express my deepest gratitude to the local organizing committee at the University of Bordeaux 1. Thanks go to the LaBRI (*Laboratoire Bordelais de Recherche en Informatique* – computer science laboratory) and to the SCRIME (*Studio de Création de de Recherche en Informatique et Musique Électroacoustique*). Thanks must also go to all the organizations that support the DAFx-07 conference.

Welcome to Bordeaux, enjoy its monuments and specialties, and of course enjoy DAFx-07!

Sylvain Marchand
DAFx-07 Conference Chair
Associate Professor,
LaBRI – University of Bordeaux 1

DAFx-07 Local Organizing Committee

LaBRI[1], SCRIME[2], Université Bordeaux 1

Sylvain	Marchand	(Conference Chair)
Annick	Alexaline	
Myriam	Desainte-Catherine	
Christian	Eloy	(Concert Organization)
Pierre	Hanna	
Joseph	Larralde	
Guillaume	Meurisse	
Joan	Mouba	
Martin	Raspaud	
Matthias	Robine	
Robert	Strandh	

[1] *Laboratoire Bordelais de Recherche en Informatique*
[2] *Studio de Création et de Recherche en Informatique et Musique Électroacoustique*

DAFx Conferences

DAFx is an acronym for **digital audio effects**, and nowadays refers to the name of the International Conference on Digital Audio Effects and to the related book "DAFX – Digital Audio Effects" edited by Udo Zölzer. It was initiated from a European research project for cooperation and knowledge diffusion (EU-COST-G6 "Digital Audio Effects", 1997–2001). Since then DAFx has been running on its own feet as a self-funded event. Papers of the conferences are available online at:

DAFx 1998	Barcelona, Spain	`http://www.iua.upf.es/dafx98/`
DAFx 1999	Trondheim, Norway	`http://www.notam02.no/dafx99/`
DAFx 2000	Verona, Italy	`http://profs.sci.univr.it/~dafx/`
DAFx 2001	Limerick, Ireland	`http://www.csis.ul.ie/dafx01/`
DAFx 2002	Hamburg, Germany	`http://www2.hsu-hh.de/ant/dafx2002/dafx2002.html`
DAFx 2003	London, United Kingdom	`http://www.elec.qmul.ac.uk/dafx03/`
DAFx 2004	Naples, Italy	`http://dafx04.na.infn.it/`
DAFx 2005	Madrid, Spain	`http://dafx05.ssr.upm.es/`
DAFx 2006	Montreal, Quebec, Canada	`http://www.dafx.ca/`
DAFx 2007	**Bordeaux, France**	`http://www.dafx.u-bordeaux.fr/`
DAFx 2008	Helsinki, Finland	
DAFx 2009	Como / Milan, Italy	
DAFx	general website	`http://www.dafx.de/`

DAFx Scientific Committee

DAFx-07 Program Committee

Jonathan	Abel	CCRMA, Stanford University, USA
Gerald	Beauregard	muvee Technologies, Singapore
F. Javier	Casajús	ETSI Telecomunicación – Universidad Politécnica de Madrid, Spain
Roger	Dannenberg	Carnegie Mellon University, USA
Laurent	Daudet	LAM, Université Pierre et Marie Curie (Paris VI), France
Bertrand	David	GET – Télécom Paris, France
Giovanni	De Poli	CSC, University of Padova, Italy
Carlo	Drioli	University of Verona, Italy
Georg	Essl	Deutsche Telekom Laboratories, Technical University of Berlin, Germany
Gianpaolo	Evangelista	Federico II University of Naples, Italy
Emmanuel	Favreau	Institut National de l'Audiovisuel – GRM, Paris, France
Federico	Fontana	University of Verona, Italy
Günter	Geiger	Pompeu Fabra University, Barcelona, Spain
Philippe	Guillemain	Laboratoire de Mécanique et d'Acoustique, Marseille, France
Robert	Höldrich	Institute of Electronic Music and Acoustics, Graz, Austria
Andrew	Horner	Hong Kong University of Science & Technology, Hong Kong
Jean-Marc	Jot	Creative Labs, USA
Matti	Karjalainen	Helsinki University of Technology, Finland
Antti	Kelloniemi	Panphonics Oy, Finland
Richard	Kronland-Martinet	Laboratoire de Mécanique et d'Acoustique, Marseille, France
Victor	Lazzarini	National University of Ireland, Maynooth, Ireland
Sylvain	Marchand	LaBRI, Université Bordeaux 1, France
James	McDermott	University of Limerick, Ireland
Søren	Nielsen	TC Electronic, Denmark
Nicola	Orio	IMS, University of Padova, Italy
Yann	Orlarey	GRAME, Lyon, France
Luis	Ortiz Berenguer	EUIT Telecomunicación – Universidad Politécnica de Madrid, Spain
Geoffroy	Peeters	IRCAM, Paris, France
Henri	Penttinen	Helsinki University of Technology, Finland
Laurent	Pottier	Université Jean Monnet, Saint-Étienne, France
Rudolf	Rabenstein	Erlangen–Nuremberg University, Germany
Jean-Bernard	Rault	France Télécom R&D, Rennes, France
Axel	Röbel	IRCAM, Paris, France
Jøran	Rudi	NoTAM, Oslo, Norway
Mark	Sandler	Queen Mary, University of London, United Kingdom
Lauri	Savioja	Helsinki University of Technology, Finland
Diemo	Schwarz	IRCAM, Paris, France
Xavier	Serra	Pompeu Fabra University, Barcelona, Spain
Soledad	Torres Guijarro	ETSI Telecomunicación – Universidad Politécnica de Madrid, Spain
Vesa	Välimäki	Helsinki University of Technology, Finland
Gualtiero	Volpe	University of Genova, Italy
Olivier	Warusfel	IRCAM, Paris, France
Sølvi	Ystad	Laboratoire de Mécanique et d'Acoustique, Marseille, France
Udo	Zölzer	Helmut-Schmidt University, Hamburg, Germany

Contents

Session: Audio Effects

Session: Sound Modeling and Analysis 1

Poster Session 1

Session: Sound Modeling and Analysis 2

Keynote 1

Session: Room Acoustics and Hardware Implementation

Session: Sound Spatialization and Localization

Poster Session 2

Session: Audio Coding

Session: Physical Modeling

TIME-SCALING OF AUDIO SIGNALS WITH MUTI-SCALE GABOR ANALYSIS

Olivier Derrien

ISITV - Université du Sud Toulon-Var
Av. G. Pompidou, BP 56 F-83162, La Valette du Var Cedex, France
olivier.derrien@univ-tln.fr

ABSTRACT

The phase vocoder is a standard frequency domain time-scaling technique suitable for polyphonic audio, but it generates annoying artifacts called phasiness, or loss of presence, and transient smearing, especially for high values of the time-scale parameter. In this paper, a new time-scaling algorithm for polyphonic audio signals is described. It uses a multi-scale Gabor analysis for low-frequency content and a vocoder with phase-locking on transients for the residual signal and for high-frequency content. Compared to a phase-locking vocoder alone, our method significantly reduces both phasiness and transient smearing, especially for high values of the time-scale parameter. For time-contraction (time-scale parameters lower that one), the results seem to be more signal-dependant.

1. INTRODUCTION

Time-scale modification of audio aims at changing the playback rate of a recorded signal without altering its frequency content, i.e. pitch and timbre. For instance, time-scaling is useful for electronic music composers who want to synchronize musical samples in order to produce a coherent output signal. A time-scaling effect consists either of a speeding up, called time-contraction, either of a slowing-down, called time-stretching.

Time-scaling techniques can be roughly classified in two categories: time-domain and frequency-domain. Time domain algorithms, typically synchronized overlap-add (SOLA) [1], are usually very efficient and can produce high-quality audio output, but only when applied to quasi-periodic signals, speech for instance. In the case of more complex audio content, like polyphonic music, time-domain methods perform poorly. Frequency-domain methods, typically phase vocoder, can be applied to both quasi-periodic and complex audio signals, still with major drawbacks: a higher computational cost and annoying artifacts in the output signal. These artifacts are usually known as transient smearing and phasiness. Transient smearing consists of a loss of percussiveness, and phasiness can be compared to an artificial reverberation effect, or a loss of presence. In fact, these two aspects are related: smooth attacks and a notable reverberation are often associated with a long distance between the source and the listener.

The phase vocoder was introduced by Flanagan *et al.* [2] in 1966, but a considerable amount of studies have focused on improving the vocoder audio quality. Laroche *et al.* [3] explained the phasiness effect by a loss of phase consistency across the vocoder channels, and developed a phase locking technique to restore partially this coherence. This method can be considered as an improvement of the method by Puckette [4]. A constant frame-rate version of the phase vocoder was proposed by Bonada [5]. Different phase-locking techniques on transients location were published by Duxbury *et al.* [6], and by Röbel [7]. Dorran *et al.* [8] also proposed a method for maintaining phase coherence between vocoder channels, but only for moderate time-scale factors. A real-time software implementation was recently described by Karrer *et al.* [9] and an hybrid approach mixing SOLA and phase vocoder was proposed by Dorran *et al.* [10]. Despite significant improvements, some artifacts remain.

Sinusoidal modeling is another class of frequency techniques suitable for time-scaling of audio. More precisely, sinusoidal modeling is commonly used in parametric audio/speech coding at low bitrate, for instance in MPEG-4 HILN [11]. The output of the synthesis module can be easily time-scaled, but the overall signal quality is poor (typically between 1/5 and 2/5 on the MOS scale [12]). Surprisingly, sinusoidal modeling for high-quality time-scaling of audio signals have received very few attention so far. In this paper, we describe a new time-scaling technique based on a multi-scale sinusoidal analysis. We also propose a hybrid time-scaling algorithm combining this method to a phase-locking vocoder, and show that both transient smearing and phasiness are significantly reduced compared to the phase-locking vocoder alone.

The paper is organized as follows: section 2 provides an overview of the phase vocoder with phase-locking on transients. In section 3, the focus is on our multi-scale sinusoidal analysis and its application to time-scaling of audio signals. Section 4 describes the hybrid algorithm and a comparison with the vocoder alone is given. Section 5 concludes.

2. PHASE VOCODER TIME-SCALING

In this section, we describe the phase vocoder that we have implemented as a reference method. Although it might not be considered as a top-level vocoder, the phase-locking technique significantly improves the signal quality compared to a basic phase vocoder.

2.1. Phase vocoder basics

In a Discrete Fourier Transform (DFT) implementation of the phase vocoder, the audio signal x is analyzed with a N-point DFT and a R_a hop-size. Thus, two successive analysis intervals overlap by $N - R_a$ samples. X are the DFT coefficients:

$$X(u,k) = \sum_{n=0}^{N-1} w_a[n]\, x[n + uR_a - N/2]\, e^{-j2\pi \frac{kn}{N}} \qquad (1)$$

w_a is the analysis window, $u \in \mathbb{N}$ is the analysis interval index, and $k \in [0 \cdots N-1]$ is a frequency index. Each value of index k corresponds to a vocoder channel. uR_a are the analysis time-instants.

Between the analysis and the synthesis stage, the signal is modified in the DFT domain. These modifications will be explained further on. Y denote the modified coefficients. The synthesis involves an iDFT:

$$y_u[n] = \frac{1}{N} \sum_{k=0}^{N-1} Y(u,k)\, e^{j2\pi \frac{kn}{N}} \tag{2}$$

for $n \in [0 \cdots N-1]$, $y_u[n] = 0$ otherwise. The final output signal y is obtained with an overlap-add operation:

$$y[n] = \sum_u y_u[n - uR_s + N/2] \tag{3}$$

R_s is the synthesis hop-size, and uR_s are the synthesis time-instants. The time-scale factor is: $\alpha = \frac{R_s}{R_a}$.

In the absence of modification, i.e. $\alpha = 1$, one simply define $Y(u,k) = X(u,k)$, and the output signal y is similar to x, depending on the analysis window w_a. For instance, a Hanning window ensures the perfect reconstruction. When $\alpha \neq 1$, the amplitude of the DFT coefficients is preserved: $|Y(u,k)| = |X(u,k)|$, but the phases are modified according to the following method.

At the first analysis/synthesis instant, we initialize:

$$\angle Y(0,k) = \angle X(0,k) \tag{4}$$

Other initializations are possible, but this one suits any value of the time-scale factor α [3]. If the signal in each channel were a single pure sine of frequency $2\pi \frac{k}{N}$, the modified phase $\angle Y(u,k)$ could be computed for every u according to the phase propagation formula from instant $(u-1)R_s$ to uR_s:

$$\angle Y(u,k) = \angle Y(u-1,k) + R_s\, 2\pi \frac{k}{N} \tag{5}$$

However, the signal is not a single pure sine, and the DFT coefficients exhibit a phase increment. The analysis phase increment can be measured:

$$\Phi_a(u,k) = \angle X(u,k) - \left(\angle X(u-1,k) + R_a\, 2\pi \frac{k}{N} \right) \tag{6}$$

The synthesis phase increment is $\Phi_s(u,k) = \alpha\ \mathrm{PD}\left(\Phi_a(u,k)\right)$, where PD is the principal determination of an angle. Finally, the complete phase propagation formula is:

$$\angle Y(u,k) = \angle Y(u-1,k) + R_s\, 2\pi \frac{k}{N} + \Phi_s(u,k) \tag{7}$$

2.2. Phase locking at transient locations

Computing the synthesis phases according to the phase propagation formula (7) ensures the horizontal phase coherence inside each channel. But the vertical phase coherence between channels is lost, which causes transient smearing and phasiness [3].

Obviously, both horizontal and vertical phase coherence can not be achieved at any time and for every channel, but many researches have focused on finding a good balance between the two. Recent studies have shown that the vertical coherence is particularly crucial at transient locations [5, 6, 7]. Thus, preserving the horizontal phase coherence on stationary regions and forcing vertical coherence at transients, for instance by resetting the synthesis phases, also called phase-locking, seems to be a good solution, but it requires a transient detection algorithm. However, resetting the phases on high-energy stationary partials coming though a transient region must be avoided, because the signal energy suddenly collapses in front of the transient. In solution proposed by Duxbury *et al* [6], only the stationary regions are time-scaled, whilst the phase is locked and the time-scale factor is forced to be one at transients. Despite local variances in time-scaling factor, rhythm is maintained globally. In the algorithm by Röbel [7], both the transient detection and the transient processing algorithms operate on the level of frequency channels: the transient detection process classifies the channels in transient/non-transient content, and the synthesis phase is reset only in non-transient channels. Furthermore, the phase reset is performed only when the transient is located close to the center of the analysis interval, so there is no need to force the time-scale factor to be one.

2.3. Implementation details

The phase-locking vocoder that we implemented as a reference technique is close to algorithm proposed by Röbel.

The choice of the DFT size N is a trade-off between frequency-distortion on low-frequency partials and transient smearing: a high value for N gives a good ability to reproduce low-frequency partials but generates a considerable transient smearing effect. At $f_s = 44100$ Hz, 2048 samples (46.5 ms) seems to be a good value. The choice of the analysis hop-size R_a is a trade-off between high-frequency buzzy artifacts due to the synthesis overlap-add, and a coarse discretization step for the time-scaling factor α: a high value for R_a produces a high quality synthesis, but as $R_s \in \mathbb{N}^*$, the possible time-scaling factors are $\alpha = \frac{k}{R_a}, k \in \mathbb{N}^*$. If we set $R_a = 8$ samples at $f_s = 44100$ Hz, synthesis artifacts are clearly perceptible for $\alpha = 1.5$. $R_a = 4$ samples seems to be a good value. Possible time-scaling factors are then 0.25, 0.5, 0.75, 1, 1.25, 1.5 etc.

The transient detection algorithm is based on the energy evolution in frequency subband, whilst Röbel proposes a more complex criterion (center of gravity of the instantaneous energy in each subband and each analysis interval). The signal, sampled at $f_s = 44100$ Hz, is analyzed with a 512-point DFT and a 75% overlap, to preserve a good time-resolution. In each subband, when the energy increases by more than 10 dB, the subband is marked. In each analysis interval, if the number of marked subbands exceeds half of the total number of subbands, we decide that a transient is located at the center of the interval. In the vocoder, the synthesis phases are reset only on marked subbands at transient-marked locations. On figure 1, we plot the spectrogram of a glockenspiel signal (from the SQAM database [13]), and phase-reset locations. One can observe that the high energy partials are preserved.

3. MULTI-SCALE GABOR ANALYSIS

In this section, we present the multi-scale sinusoidal analysis that we use in our time-scaling algorithm. First, we describe the redundant time-frequency dictionary composed of Gabor waveforms and the decomposition method which is basically a modified version on the Matching Pursuit algorithm. Then, we explain how the time-scaling operation is applied to each atom.

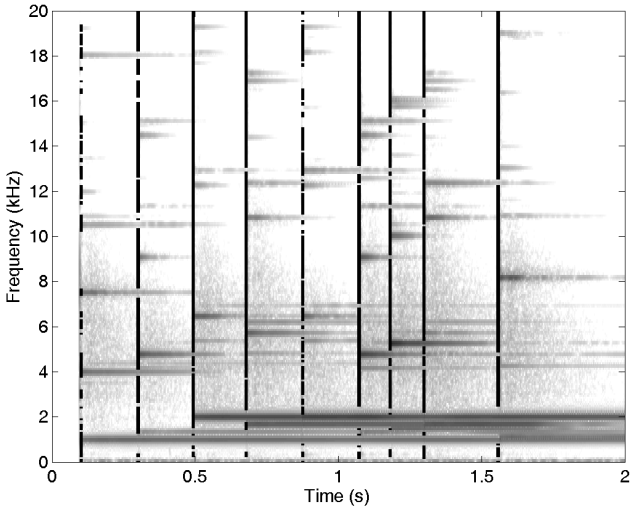


Figure 1: *Spectrogram of a glockenspiel signal (gray) and phase-reset locations (black).*

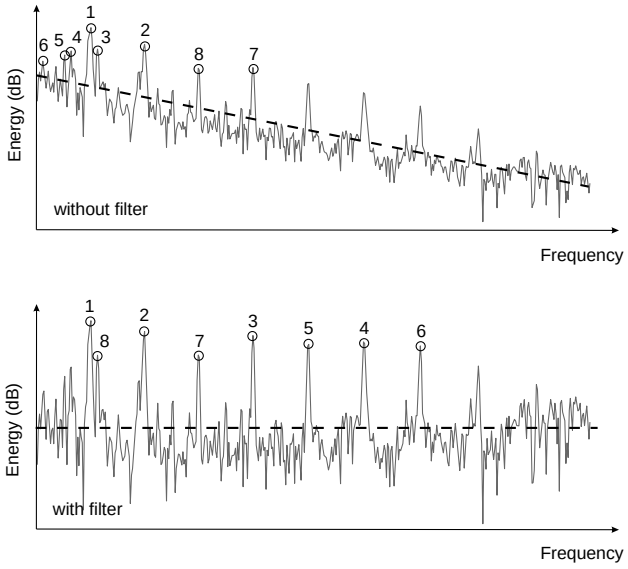


Figure 2: *Example of components selection order with Matching Pursuit in the frequency domain, with and without the adaptive filter.*

3.1. Time-frequency dictionary

The underlying signal model is a linear combination of time-frequency waveforms g plus a residual signal r:

$$x[n] = \sum_i a_i g_{\lambda_i}[n] + r[n] \tag{8}$$

$a_i g_{\lambda_i}[n]$ are called atoms. $g_\lambda[n]$ are complex Gabor waveforms [14], defined by:

$$g_\lambda[n] = \gamma(s)\, h_g\left(\frac{n-p}{s}\right)\, e^{j2\pi\nu n}, \quad \lambda = \{s, p, \nu\} \tag{9}$$

s is the time-scale factor, p the translation parameter and ν the modulation frequency. $h_g(t)$ is the amplitude function and $\gamma(s)$ is a normalization factor, depending on s. The dictionary is the over-complete set of all possible waveforms. In a classic Gabor dictionary, $h_g(t)$ is a Gaussian function. For this application, we rather use a Hanning window, which is a compactly-supported function:

$$h_g(t) = (1 + \cos(2\pi t))\ .\ \mathbf{1}_{[0,1[}(t) \tag{10}$$

Parameters are discretized in the following way:

$$s = 2^q, \quad i \in \{q_{\min} \cdots q_{\max}\} \tag{11}$$

$$p = uR_g, \quad u \in \mathbb{N} \tag{12}$$

$$\nu = \frac{k}{s}, \quad k \in \{1 \cdots s-1\} \tag{13}$$

R_g, the hop-size, is set to $2^{q_{\min}-1}$ and does not depend on the time-scale. This differs from the usual discretization in Gabor dictionaries, where the hop-size depends on the time-scale factor (usually $p = u\frac{s}{2}$). In other words, the overlap factor increases with the time-scale in our dictionary and equals 50% for $s = 2^{q_{\min}}$, whilst the overlap factor equals 50% for all time-scales in the usual discretization. This choice was made in order to limit the phase rotation between consecutive atoms, which is crucial in the context of time-scaling.

3.2. Decomposition algorithm

The decomposition algorithm determines a suitable set of index λ_i under a matching constraint, usually related to the energy of the residual signal r. The decomposition is performed on a frame-by-frame basis. Thus, only a limited subset of waveforms is considered in each frame. The time-segmentation stage is very similar to the one performed before a DFT: we use N-points intervals, with a R_a hop-size, and an analysis window w_a. We choose a set of parameters that match the Gabor dictionary: $N = 2^{q_{\max}}$, $R_a = R_g = 2^{q_{\min}-1}$ and $w_a[n] = h_g(\frac{n}{N})$. In the current frame, only the waveforms that completely overlap the analysis window are considered for the decomposition. When the same waveform is selected in different overlapping frames, which is a usual case, the final atom is computed by simply adding all the complex coefficients a_i associated to this waveform, bearing in mind the phase offset due to the translation of the analysis interval.

Our algorithm is a modified version of the iterative Matching Pursuit (MP) proposed by Mallat *et al.* [15]. The MP can be summarized as follows: at the beginning, the residual signal is equal to the signal itself. At each step, an atom is subtracted from the residual signal. This atom is co-linear to the waveform that maximizes the modulus of the inner-product with the residual signal. The decomposition is stopped when a matching criterion is smaller that a pre-defined threshold. The difference between the standard MP and our modified algorithm is that ours selects each atom from a filtered version of the residual signal. The filter transfer function is log-linear and computed for each frame so that the baseline of the filtered signal spectrum is approximately flat. Without this filter, the standard MP algorithm picks the most energetic component in the residual signal at each iteration. For instance, a high-energy noise component in low-frequency will be selected before a high-frequency partial with a lower energy. With the filter, the

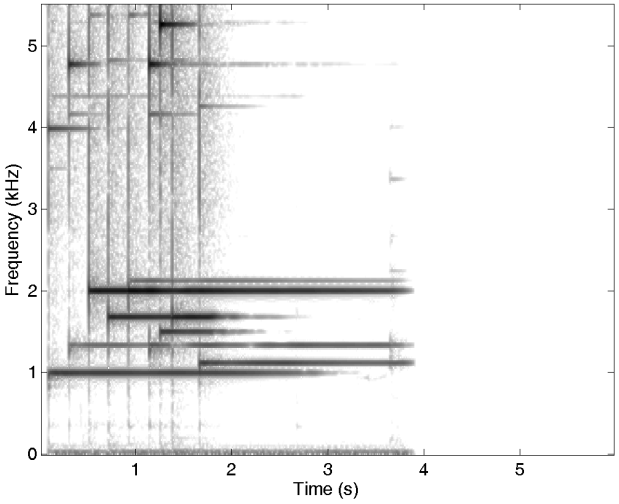


Figure 3: *Spectrogram of a glockenspiel signal.*

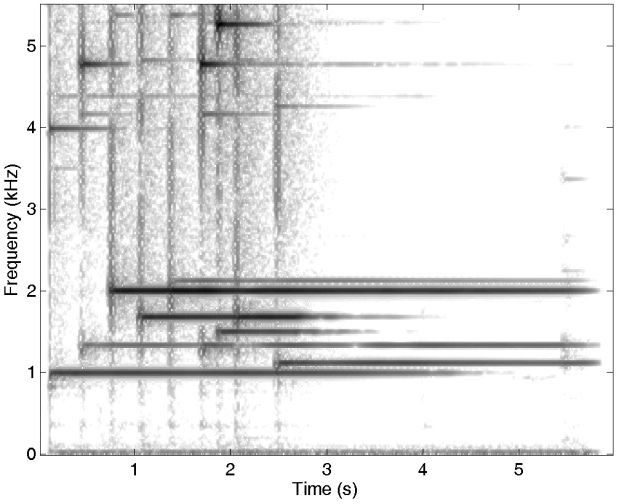


Figure 4: *Spectrogram of a glockenspiel signal time-scaled by a phase vocoder with phase-locking at transients,* $\alpha = 1.5$.

high-frequency partial is amplified and selected before the noise component. Our method improves the selection of significant partials, and leaves the noise components in the residual signal (see example on figure 2).

In the standard MP, the matching criterion is the energy of the residual signal. However, we found that combining this criterion with the correlation between the selected atom and the residual signal is more efficient. The exact description of our algorithm is as following. We denote $\tilde{x}$ and $\tilde{r}$ respectively the filtered versions of x and r, and M the matching criterion.

Initialization : set $i = 0$, $r_0 = x$ and $\tilde{r}_0 = \tilde{x}$

while $M(\tilde{r}_i) > \varepsilon$

Compute $\forall \lambda$ the inner-product $\langle \tilde{r}_i, g_\lambda \rangle$

Select the best waveform index:
$\lambda_i = \text{Argmax}_\lambda \, |\langle \tilde{r}_i, g_\lambda \rangle|$

Subtract the corresponding atom:
$a_i = \langle r_i, g_{\lambda_i} \rangle$
$\tilde{a}_i = \langle \tilde{r}_i, g_{\lambda_i} \rangle$
$r_{i+1} = r_i - a_i \, g_{\lambda_i}$
$\tilde{r}_{i+1} = \tilde{r}_i - \tilde{a}_i \, g_{\lambda_i}$

Increment the waveform index: $i = i + 1$

end

3.3. Atoms time-scaling

Assuming that the residual signal is not perceptually significant, the time-scaling operation can be achieved by scaling the linear combination of time-frequency waveforms, i.e. by scaling each atom. The basic rule for scaling an atom is as follows: on stationary regions, the time-scale parameter s and the translation parameter p are scaled, whilst the modulation frequency ν remains unchanged. When the center of an atom is located on a transient, the atom is not scaled in order to preserve the time-envelope of the transient.

Concerning amplitude and phase of the modified atoms, we propose the following rule: for the current atom, if no previous overlapping atom with the same frequency exists in the decomposition, the amplitude and phase are kept unchanged. Otherwise, the amplitude is kept unchanged and the phase propagation formula is applied.

More precisely: first, in the decomposition formula (8), the atoms are classified according to:

1. increasing translation parameter p
2. decreasing energy $|a_i|^2$

Then, for each atom $a_i g_{(s_i,p_i,\nu_i)}$, the modified atom $a'_i g_{(s'_i,p'_i,\nu_i)}$ is computed as follows. Concerning the waveform parameters:

- if p_i is located on a transient and if ν_i is in a transient-marked subband, $s'_i = s_i$ and $p'_i = p_i$.
- else, $s'_i = \alpha s_i$ and $p'_i = \alpha p_i$.

Concerning the complex coefficient, the amplitude is preserved: $|a'_i| = |a_i|$, and for the phase:

- if a previous overlapping atom $a_j g_{(s_j,p_j,\nu_j)}$ with $\nu_j = \nu_i$ exists in the decomposition, the phase is set according to the phase-propagation formula. The phase increment between atoms j and i is:

$$\Phi_{ji} = \angle a_i - (\angle a_j + (p_i - p_j)2\pi\nu_i) \qquad (14)$$

and the modified phase is:

$$\angle a'_i = \angle a'_j + \alpha(p_i - p_j)2\pi\nu_i + \alpha \text{PD}\left(\Phi_{ji}\right) \qquad (15)$$

- else $\angle a'_i = \angle a_i$.

With this method, no explicit phase-locking is necessary on transient locations.

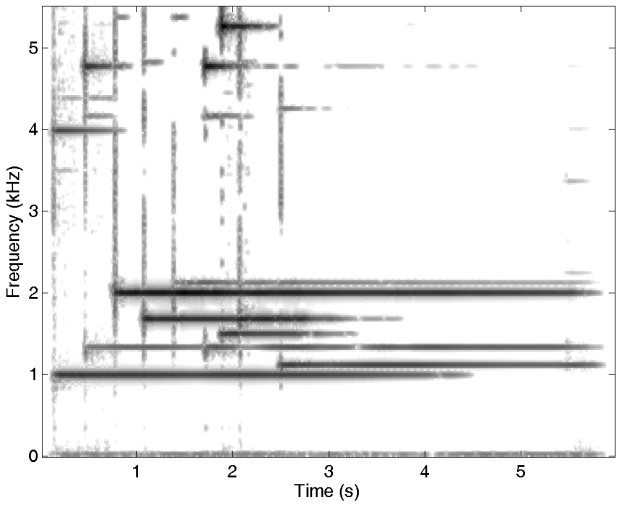


Figure 5: *Spectrogram of a glockenspiel signal time-scaled by Gabor analysis,* $\alpha = 1.5$.

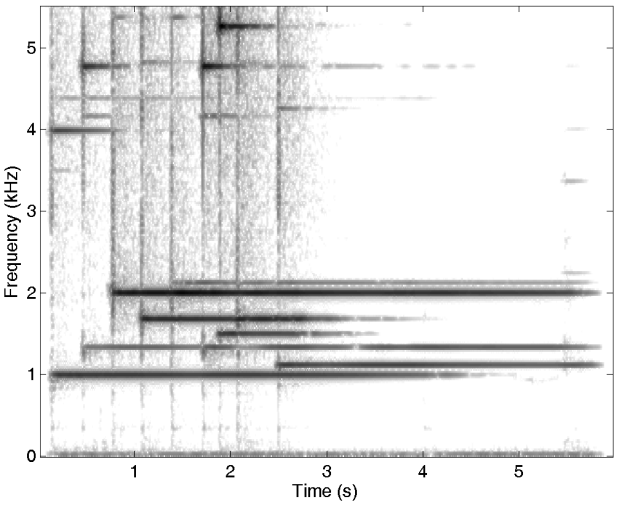


Figure 6: *Spectrogram of a glockenspiel signal time-scaled by the hybrid method,* $\alpha = 1.5$.

3.4. Implementation details and first results

According to Mallat [15], the complexity of the Matching Pursuit is similar to the one of the FFT i.e. $\mathcal{O}\left(N \log(N)\right)$, when implemented in a efficient way. Practically, one can observe that the MP implementation is significantly more complex that the FFT. In our experiments, we chose to downsample the audio signal from 44100 Hz to 11025 Hz in order to limit the complexity.

The length of analysis intervals is set to $N = 1024$ samples (93 ms), which is twice the length of the phase vocoder analysis intervals. Thus, the theoretical frequency-resolution is twice better. The hop-size is set to $R_g = 64$ samples. We get $i_{\max} = 10$ and $i_{\min} = 7$. The theoretical time-resolution is 11.5 ms. However, the practical time and frequency resolution strongly depend on the decomposition algorithm.

We have tested both methods, phase vocoder and Gabor analysis, through informal listening test on real polyphonic music signals, for different time-scale factors. Concerning the phase vocoder, the main conclusions are:

- The phase locking technique significantly reduces artifacts,
- But perceptible phasiness and transient smearing effects still appear.

and for Gabor analysis:

- On downsampled signal, this method generates fewer artifacts than the phase-locking vocoder,
- But noise components are missing.

As a graphical illustration, we show spectrograms of a glockenspiel signal. Figure 3 corresponds to the original (unprocessed) signal. On figure 4, the signal is time-scaled with the phase-locking vocoder for $\alpha = 1.5$. The audible transient smearing effect is visually noticeable on this plot: attack regions are stretched and look granular. Otherwise, the frequency content of the original signal seems preserved. On figure 5, the signal is time-scaled with the Gabor analysis method, with a signal-to-residual noise around 30 dB. This corresponds to an average number of 35 atoms per frame of 1024 samples (about 20 atoms in stationary regions, and about 150 to 200 atoms in transient regions). One can see that the time-smearing effect is reduced, but only high energy components are treated, and most of the noise components are left in the residual signal.

4. HYBRID TIME-SCALING

In this section, we describe our complete time-scaling method, based on both Gabor analysis and phase vocoder.

4.1. Hybrid method

The Gabor analysis method can hardly by used alone for time-scaling a full-bandwidth audio signal, because it would require a very high number of atoms per frame, possibly higher than the number of samples, and the resulting complexity would be excessive. We think that the most efficient approach consists of stopping the Gabor analysis when no significant partial is left in the residual signal. It can be achieved by downsampling the original signal and perform the Gabor analysis with a medium matching criterion. The atoms are scaled according to the algorithm decribed in the previous section. The residual signal, which contains noise components in the low-frequency band and all the high-frequency content, is scaled with a phase-locking vocoder. As there is no significant partial left in the residual signal, one can choose a higher time-resolution than when scaling the full signal. We set $N = 1024$ (23 ms) and $R_a = 4$ samples. The transient smearing effect is not contained, and no buzzy artifact is perceptible.

4.2. Final results

On figure 6, we plot the spectrogram of the glockenspiel signal scaled with our hybrid Gabor analysis/vocoder technique. One can observe that, compared to the Gabor analysis alone, the transient smearing effect is not increased and remains lower than with

vocoder alone, whilst noise components are preserved in the scaled signal.

Informal listening tests, involving 4 listeners and 4 different audio excerpts have shown that the signal quality is improved compared to the phase-locking vocoder alone. Our method significantly reduces both the transient smearing and phasiness effects: the presence effect in the scaled signal is much better than with the phase vocoder alone, especially for high values of the scaling parameter ($\alpha > 1.5$), for which the scaled signal often sounds artificial. However, when $\alpha < 1$, the phase vocoder might perform better, on some very specific audio signals.

Examples of audio signals processed with both methods for various scaling parameters can be found on the DESAM project website: http://www.tsi.enst.fr/~rbadeau/desam/spip.php?article16.

5. CONCLUSION

In this paper, a new high-quality time-scaling algorithm for polyphonic audio signals has been presented. It is based on a multi-scale Gabor analysis for low-frequency content (between 0 and 5.5 kHz), and on a phase-locking vocoder for high-frequency content (between 5.5 and 22 kHz) and for the residual part in the low-frequency band. In the time stretching context, i.e. $\alpha > 1$, our method significantly reduces the two main artifacts generated by a phase vocoder: phasiness and transient smearing. The improvement is particularly interesting when the scaling parameter is high ($\alpha > 2$). However, for time-contraction, i.e. $\alpha < 1$, the results seem to be more signal-dependant.

Compared to the phase vocoder, the overall complexity of the time-scaling process is significantly higher with our method. First because the Gabor analysis is more complex than a FFT, second because our method also requires a phase vocoder for the residual signal. This makes out method unsuitable for real-time implementations for the moment.

This study proves that Gabor analysis is a valid alternative to the phase vocoder for audio time-stretching, but must be considered as preliminary. In further studies, we will extend our method to full-bandwidth signals. We will also try to define more complex rules for time-stretching the atoms, with partials tracking for instance. We also work on a more complex signal model which would not require a phase vocoder for processing the residual signal.

6. ACKNOWLEDGEMENTS

The research leading to this paper was supported by the French GIP ANR under contract ANR-06-JCJC-0027-01, Décomposition en Éléments Sonores et Applications Musicales - DESAM. See the project website: http://www.tsi.enst.fr/~rbadeau/desam/ for more details.

7. REFERENCES

[1] S. Roucos and A. M. Wilgus, "High quality time-scale modification of speech," in *Proc. of the IEEE ICASSP International Conference on Acoustics, Speech and Signal Processing*, 1985.

[2] J. L. Flanagan and R. M. Golden, "Phase vocoder," *The Bell System Technical Journal*, vol. 45, pp. 1493–1509, November 1966.

[3] J. Laroche and M. Dolson, "Improved phase vocoder time-scale modification of audio," *IEEE Transactions on Speech and Audio Processing*, vol. 7, no. 3, pp. 323–332, May 1999.

[4] M. S. Puckette, "Phase-locked vocoder," in *Proc. of the IEEE ASSP Workshop on Applications of Signal Processing to Audio and Acoustics, Mohonk, New-York*, 1995.

[5] J. Bonada, "Automatic technique in frequency domain for near-lossless time-scale modification of audio," in *Proc. of the ICMC International Computer Music Conference, Berlin, Germany*, 2000.

[6] C. Duxbury, M. Davies, and M. Sandler, "Improved time-scaling of musical audio using phase locking at transients," in *Proc. of the 112th Convention of the Audio Engineering Society, Munich, Germany*, 2002.

[7] A. Röbel, "A new approach to transient processing in the phase vocoder," in *Proc. of the 6th International Conference on Digital Audio Effects (DAFx-03), London, UK*, 2003.

[8] D. Dorran, E. Coyle, and R. Lawlor, "An efficient phasiness reduction technique for moderate audio time-scale modification," in *Proc. of the 7th International Conference on Digital Audio Effects (DAFx-04), Naples, Italy*, 2004.

[9] T. Karrer, E. Lee, and J. Borchers, "Phavorit: A phase vocoder for real-time interactive time-stretching," in *Proc. of the ICMC International Computer Music Conference, New-Orleans, USA*, 2006.

[10] D. Dorran, E. Coyle, and R. Lawlor, "Audio time-scale modification using a hybrid time-frequency domain approach," in *Proc. of the IEEE ASSP Workshop on Applications of Signal Processing to Audio and Acoustics, New Paltz, New-York*, 2005.

[11] International Organization for Standardization, *ISO/IEC 14496-3 (Information technology - Very low bitrate audio-visual coding - Part3: Audio)*, 1998.

[12] R. Heusdens, J. Jensen, W. Bastiaan Kleijn, V. Kot, O. A. Niamut, S. Van De Paar, N. H. Van Schijndel, and R. Vafin, "Bit-rate scalable intraframe sinusoïdal audio coding based on rate-distortion optimization," *Journal of the Audio Engineering Society*, vol. 54, no. 3, pp. 167–188, March 2006.

[13] EBU SQAM, "Sound quality assessment material recordings for subjective tests," Compact Disc, 1998.

[14] N. Delprat, B. Escudie, P. Guillemain, R. Kronland-Martinet, P. Tchamitchian, and P. Torresani, "Asymptotic wavelets and gabor analysis: Extraction of instantaneous frequencies," *IEEE Transactions on Information Theory*, vol. 38, no. 2, pp. 644–664, March 1992.

[15] S. G. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3397–3415, December 1993.

REAL-TIME PITCH-SHIFTING OF MUSICAL SIGNALS BY A TIME-VARYING FACTOR USING NORMALIZED FILTERED CORRELATION TIME-SCALE MODIFICATION (NFC-TSM)

Azadeh Haghparast, Henri Penttinen, and Vesa Välimäki

Lab. of Acoustics and Audio Signal Processing
Helsinki University of Technology, Espoo, Finland
azadeh.haghparast@tkk.fi

ABSTRACT

This paper presents a high-quality real-time pitch-shifting algorithm with a time-varying factor for monophonic audio and musical signals. The pitch-shifting algorithm is based on the resampling and time-scale modification method. A new time-scale modification method has been developed which is called the Normalized Filtered Correlation Time-Scale Modification (NFC-TSM) method. It uses a ring buffer for time-scaling. The best splicing point is searched in the normalized low-pass filtered signal using the Average Magnitude Difference Function (AMDF). The new method results in low-latency and high-quality pitch-shifting of musical signals.

1. INTRODUCTION

High-quality techniques for pitch-shifting of audio and musical signals have received a lot of attention recently. In multi-track audio recording and mixing, pitch-shifting is used to match the pitches of two recorded digital audio clips [1]. Real-time pitch-shifting algorithms can be used for performing deejays [2]. In music industry, pitch-shifting is used in sampling synthesizers, sound effects for Karaoke systems [3, 4], and other musical effects.

In general, pitch-shifting algorithms can be divided into two categories; time-domain and frequency-domain techniques [5]. Time-domain techniques are simple and fast, and work fine for periodic and quasi-periodic signals. However, their quality is not good for signals which contain a lot of non-harmonic components. On the other hand, frequency-domain algorithms are more suitable for complex signals, but the price of the high-quality is the computational complexity. Additionally, frequency-domain pitch-shifting algorithms call for large delays and, thus, are not appropriate for real-time applications. Frequency-domain algorithms are usually based on the phase-vocoder [6, 7]. In the phase-vocoder technique, first the signal is converted to its frequency-domain representation using a short-time Fourier transform (STFT). After modification of the frequency-domain parameters according to the pitch-shifting factor, the signal is converted back to its time-domain waveform.

The standard time-domain pitch-shifting algorithms, commonly used in commercial applications, are based on resampling and time-scale modification [3, 8, 5]. In the standard time-domain pitch-shifting technique, for pitch-shifting the signal by a factor of α, the input signal is first resampled by a resampling factor equal to $1/\alpha$. Since resampling changes the length of the signal, a time-scale modification method is used to preserve the time duration of the original signal. The time duration of the resampled signal should be scaled by a factor equal to α. Figure 1 shows the block diagram of a time-domain pitch-shifting technique using resampling and time-scale modification. In this figure, $f_{\mathrm{s,org}}$ and $f_{\mathrm{s,replay}}$ are the sampling frequency of the original audio and that of the pitch-shifted signal, respectively.

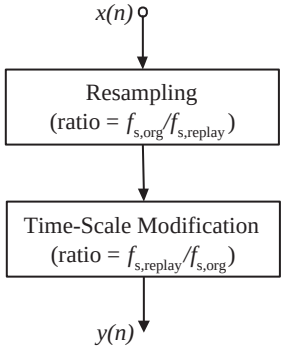


Figure 1: *Block diagram of the pitch-shifting method based on the resampling and time-scale modification.*

In general, there are two methods to implement the time-scale modification in the time domain. One of them is the ring buffer technique, which has serious quality problems [9, 10]. The other one is the overlap and add technique [5] which has many variations, developed to improve the quality of the output signal and reduce the computational complexity [1, 4, 11]. In the overlap and add method, the input signal is divided into overlapping segments which are shifted with respect to each other according to the time-scaling factor. Finally, they are added to each other to form the output signal [5]. Since overlapping and adding segments at any point breaks the continuity of the pitch and changes the spectral characteristics of the signal, the two overlapping segments have to be synchronized and cross-faded at the point of highest similarity. Several developments to find the maximum similarity points resulted in the different variations of the overlap and add method [1, 4, 11]. Some of these techniques are the synchronized overlap and add method (SOLA) [12, 13], the pitch synchronous overlap and add method (PSOLA) [5, 12, 14, 15], the waveform similarity overlap and add method (WSOLA) [16], and the global and local search time-scale modification (GLS-TSM) [17]. Recently, a comparison of the time-domain time-scale modification techniques has been presented in [18].

In this paper, a pitch-shifting method by a time-varying factor for monophonic musical tones is presented. The pitch-shifting algorithm is a standard time-domain pitch-shifting algorithm. A

new time-scale modification algorithm has been designed which is called the Normalized Filtered Correlation Time-Scale Modification (NFC-TSM). The proposed time-scale modification technique enables the real-time pitch-shifting of the input signal. Moreover, the pitch of the input signal can be scaled continuously. The pitch-shifting factor can have large amounts with no serious defects in the quality of the pitch-shifted signal. The number of clicks in the pitch-shifted signal have been reduced comparing to the previous methods. A patent application has been submitted regarding this method.

In the following, first the general description of the pitch-shifting algorithm is presented. Then, the resampling process by a time-varying factor is explained. The NFC-TSM method is described next. Setting the parameters of the algorithm is discussed. Finally, the results are presented.

2. GENERAL DESCRIPTION OF THE ALGORITHM

The proposed pitch-shifting algorithm is based on the resampling and time-scale modification method. In this technique, the audio or musical signal is assumed to be periodic or semi-periodic. Moreover, there exists one pitch-shifting factor per input sample. Pitch-shifting is carried out by resampling the signal according to the pitch-shifting factor. Meanwhile, time-scale modification is performed whenever needed to preserve the duration of the original signal. In this work, to modify the time-scale of the signal, the novel NFC-TSM algorithm is used. Note that in the presented pitch-shifting algorithm, the resampling and time-scale modification operations are incorporated and can not be separated as such.

The pitch-shifting algorithm is performed on the audio signal that is stored in a ring buffer. A ring buffer, which is also called a circular buffer, is a portion of memory of fixed size into which new data is overwritten at its beginning when it is full. Two pointers to the ring buffer are defined: the input pointer and the output pointer. The input pointer is used to write into the ring buffer. That is, it is incremented by one when one input sample is received and written in the ring buffer. The output pointer is defined for resampling the audio signal in the ring buffer.

The rates at which the input and output pointers move in the ring buffer are different due to the resampling process. The rate of the input pointer is fixed, and equal to the rate of the sampling rate of the input signal, e.g. 44100 samples/sec. The rate at which the output pointer moves depends on the pitch-shifting factor. Different speeds of the input and output pointers result in their collision. After collision, either of the pointers may pass the other one. This causes discontinuity in the output signal which, in turn, results in audible artefact in the pitch-shifted signal.

In order to avoid the collision between the two pointers, the output pointer should jump backward and forward in the ring buffer to remain behind the input pointer. The best location for the output pointer to jump to, also referred to as the best splicing point, is searched using the NFC-TSM technique. In this method, the best splicing point is searched in the normalized low-pass filtered version of the ring buffer using Average Magnitude Difference Function (AMDF) as the correlation function. After having found the best splicing point, the two segments are joined to each other using a linear cross-fading function.

Next, the resampling by a time-varying factor and the NFC-TSM algorithm are explained in detail.

2.1. Resampling by a Time-Varying Factor

Resampling is the process of interpolating a signal at non-integer multiples of the sampling period. Theoretically, it can be stated as follows: first the original signal is reconstructed from a set of samples. In the next step, it is resampled at the desired locations. In practice, these two stages can be combined so that we need to find the values of the signal at the desired locations. The process of finding the signal values at arbitrary time instants from a set of samples is called interpolation. There are a variety of interpolators [19], such as truncated sinc [3, 8], linear interpolator [20], Lagrange interpolator [21], and spline interpolator [22]. In this algorithm, truncated sinc has been used to resample a signal, since it performs well for signals with rich high-frequency content.

In order to resample the digital audio stored in the ring buffer, for every input sample, the resampling factor is determined. The resampling factor is equal to the reciprocal of the pitch-shifting factor. This way the time instant at which the value of the signal is to be interpolated is found. Then, the truncated-sinc function is lined up with its peak at this time instant. The signal samples on both sides of the interpolation point are multiplied by the corresponding sinc function values and summed up to produce the pitch-shifted sample value.

The time instant at which the sinc function is lined up is pointed to by the output pointer. The speed of the output pointer depends on the resampling factor and, in turn, pitch-shifting factor. Figure 2 shows how the resampling process influences the speed of the output pointer. In Figure 2 (a), the signal is pitch-shifted down and the output pointer moves slower than the input pointer. In contrast, Figure 2 (b) shows a case in which the signal is pitch-shifted up and the speed of the output pointer is faster than the input pointer.

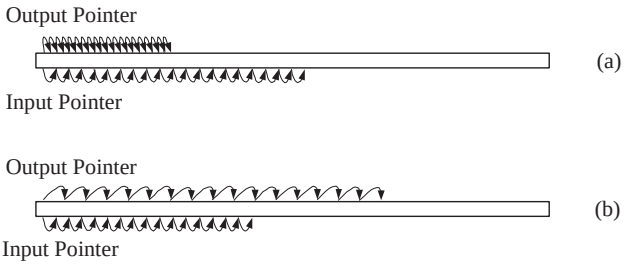


Figure 2: *Effect of resampling process on the speed of the output pointer, (a) signal is pitch-shifted down, the output pointer moves slower than the input pointer, (b) signal is pitch-shifted up, the output pointer moves faster than the input pointer.*

An important issue regarding interpolation is that the distance between the input pointer and the output pointer cannot be less than half the length of the sinc interpolator. Otherwise, the discontinuity in the sample set will result in interpolation errors.

2.2. Normalized Filtered Correlation Time-Scale Modification (NFC-TSM)

The main idea of the time-scale modification method was taken from the ring buffer method presented by Francis Lee in 1972 [10], which is based on discarding and repeating some segments of the audio signal to compress and expand the length of the signal, respectively. The conventional ring buffer technique results in audi-

ble artifacts, since the periodicity of the signal is broken and also amplitude discontinuities occur at the splicing points.

As discussed, the resampling process makes the output pointer move at a different speed from the input pointer. Different speeds of the pointers moving around a fixed-length buffer cause them to collide at some locations in the ring buffer occasionally. Collision of the input and output pointers in the ring buffer results in discontinuity in the time-scaled version of the resampled signal, which is heard as a click. To avoid this problem, in the proposed algorithm, the output pointer is handled in such a way that it never collides with the input pointer. Moreover, to maintain the time evolution of the signal, the output pointer should always keep the pace with the input pointer. This way, changes in the amplitude and frequency of the signal are followed sufficiently. Therefore, for every sample in the input, the distance between the input and output pointers is measured, d. If it is longer than, or equal to, a maximum allowed distance $d_{\max}$, the output pointer has to jump forward, behind the input pointer. On the other hand, if this distance is shorter than, or equal to, a minimum allowed distance $d_{\min}$, it is possible that soon the output pointer collides with the input pointer. Hence, it is required that the output pointer jumps backward in the ring buffer and continues resampling the signal from this point. This process has been shown in Figure 3.

Figure 3 (a) shows the case in which the input signal is pitch-shifted down and the distance between the pointers increases. Therefore, the output pointer hops forward. Figure 3 (b) shows the opposite case, when the signal is pitch-shifted up and the output pointer should jump backward to avoid collision. It is important to remember that the hop size cannot be very long, since we are aiming at following the changes in the signal.

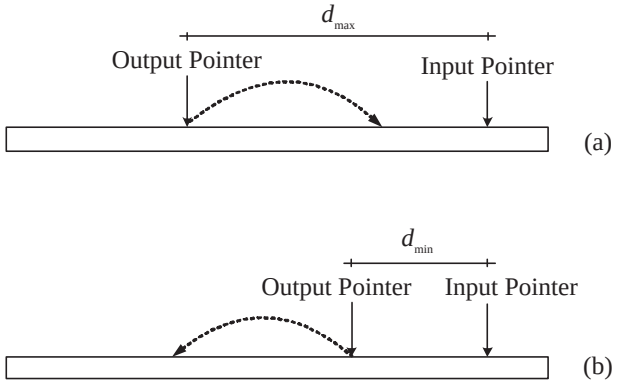


Figure 3: *Behavior of the output pointer with respect to its distance from the input pointer, d, (a) when $d > d_{\max}$, the output pointer jumps forward, (b) when $d < d_{\min}$, the output pointer jumps backward.*

The hop of the output pointer to the new location will break the periodicity of the signal. Therefore, it is required to search for the best point for the output pointer to jump so that the periodicity of the signal is maintained. This search is performed in another ring buffer of the same size as the main ring buffer. However, this second ring buffer contains the normalized low-pass filtered version of the signal stored in the main ring buffer. The best splicing point is, then, searched in the normalized low-pass filtered version of the signal using AMDF. The AMDF of two frames of length L in the signal $x(n)$ is defined as

$$D(m) = \sum_{k=0}^{L-1} |x(k+m) - x(k)|, \tag{1}$$

where m is the time lag between two frames. The maximum similarity point in the search region is the point at which AMDF is minimum for the whole search region.

The reason why the normalized low-pass filtered signal has been used in the search for the best splicing point is that we aim at preserving the continuity of the signal in its lowest partials. By normalization, the effect of the signal level alteration from one period to the other period is eliminated in the search for the best match point. The other advantage of normalization is that the amplitude changes in one period of the signal are more distinguishable with respect to the original and low-pass filtered signal. Hence, it is possible to choose a short correlation window for the AMDF search and the best splicing point can be found more accurately. Figures 4 (a)-(c) show the original signal, the low-pass filtered signal, and the normalized low-pass filtered signals, respectively.

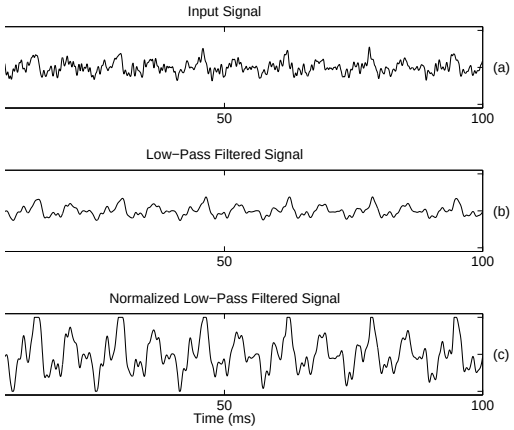


Figure 4: *(a) Original signal, (b) low-pass filtered version of the original signal, (c) normalized low-pass filtered version of the original signal.*

The choice of the correlation window and search area depends on the direction towards which the output pointer hops. Figure 5 shows two cases in which the output pointer jumps backward and forward, respectively. In Figure 5 (a), the output pointer jumps backward in the ring buffer. Therefore, the correlation window is chosen so that the output pointer points to the last sample in the correlation window. In the case the output pointer is to jump forward, the correlation window starts from the sample to which the output pointer points. This is shown in Figure 5 (b). In the figures, the best splicing points are also illustrated.

The process of finding the best splicing point is shown in Figure 6. Figure 6 (a) shows the correlation window and the search area. In Figure 6 (b), the search area has been zoomed in. The AMDF over the search area is demonstrated in Figure 6 (c). This figure shows the case the output pointer jumps forward in the ring buffer. As can be seen in the figures, the minimum AMDF results in the maximum similarity and, thus, it is highly probable that the periodicity of the signal is preserved.

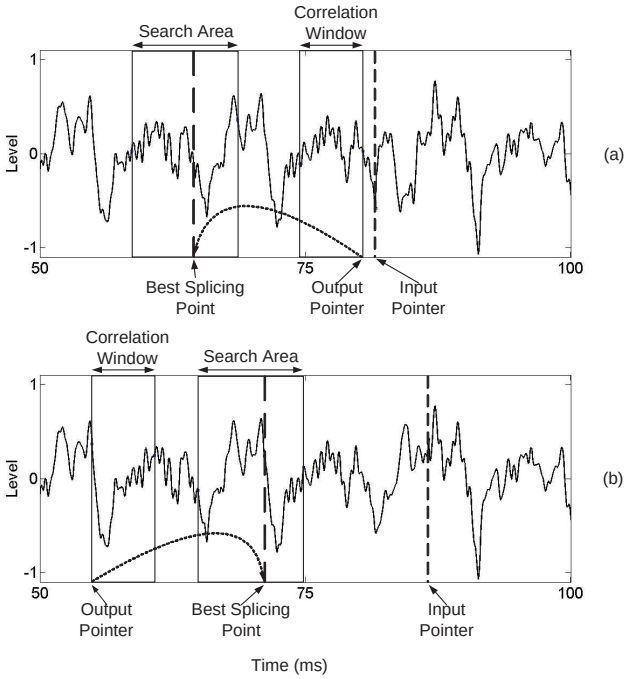


Figure 5: *Choice of the correlation window and search area in the search for the best splicing point, (a) the output pointer jumps backward, (b) the output pointer jumps forward.*

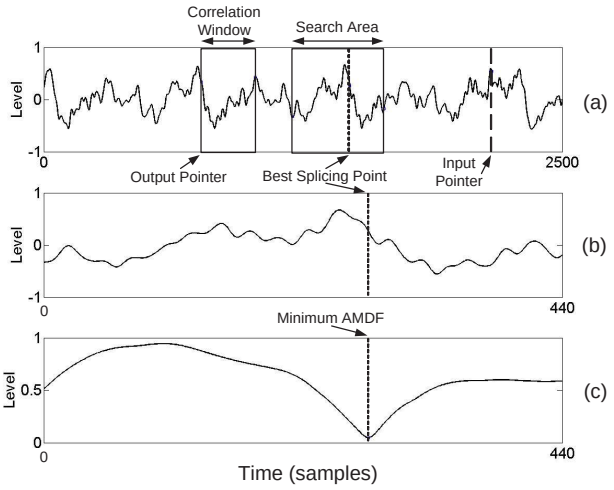


Figure 6: *Search for the best splicing point, (a) part of the ring buffer in which the correlation window and search area are placed, (b) the search area in which the best splicing point is looked for, (c) the AMDF over the search area for the shown correlation window.*

3. PARAMETER SETTING

For different input signals, the parameters of the algorithm are changed according to the period length of the lowest frequency in the frequency range of the input signal. In the following subsections, it is explained how to choose these parameters.

3.1. Maximum and Minimum Distances Between Pointers

The maximum and minimum allowed distances between the input and output pointers are essential parameters of the algorithm, which depend on the period length of the input signal. Empirically, the maximum allowed distance is chosen to be twice the longest period length of the input signal. When the distance between the output and input pointers reaches this amount, the output pointer will jump forward in the ring buffer for one period to keep the pace with the input pointer.

On the other hand, the minimum allowed distance depends on the length of the interpolator filter, the length of the cross fading region and the maximum pitch-shifting factor for upward pitch-shifting. The minimum allowed distance cannot be less than half the length of the interpolation filter. The length of the cross-fading region should be also considered. If the interpolation filter length is equal to N and the maximum pitch-shifting factor is $\alpha_{\max}$, the minimum allowed distance between input and output pointers is obtained by

$$d_{\min} = (\alpha_{\max} - 1)L_{\mathrm{CrossFade}} + \frac{N}{2}, \tag{2}$$

where $L_{\mathrm{CrossFade}}$ is the length of the cross-fading region. When the distance between the pointers reaches to $d_{\min}$, the output pointer jumps backward on the ring buffer for one period.

3.2. Lengths of Correlation Window and Search Area

The length of the correlation window L_{CorrWin}, and the length of the search area $L_{\mathrm{SearchArea}}$ depend on the period length of the lowest frequency in the frequency range of the input signal T.

$$L_{\mathrm{CorrWin}} = CT, \tag{3}$$

$$L_{\mathrm{SearchArea}} = (1 - C)T, \tag{4}$$

where C is chosen to be $3/8$ in this algorithm, although a choice of $1/4$ of one period length of the signal for the length of the correlation window contains enough information to find its right location in the period.

The search area contains all the points in the ring buffer on which the first point of the correlation window is placed and its AMDF is computed in every iteration. Therefore, if the length of the search area is selected to be $(T - L_{\mathrm{CorrWin}})$, the correlation window moves over one period of the signal. This is sufficient to find the best match point.

Another important parameter is the starting point of the search area. The choice of the search start point depends on the direction that the output pointer jumps in the ring buffer. However, it should be close enough to the input pointer, in order to follow changes in the signal. It is usually selected to be as far as one period length of the lowest frequency of the frequency range apart from the input pointer.

3.3. Other Parameters

There are a few other parameters that should be set in the algorithm. The first one is the size of the ring buffer. The ring buffer should have space for about four periods of the signal. Therefore, it can be chosen according to the longest period in the frequency range of the input signal.

The other parameter is the cut-off frequency of the low-pass filter, which is not very critical. However, it has to be greater than the highest fundamental frequency in the frequency range of the input signal. For example, if the fundamental frequency of the input signal ranges between 300 to 900 Hz, the cut-off frequency of the low-pass filter should be greater than 900 Hz.

The normalization is performed by multiplying the low-pass filtered signal by the reciprocal of its temporal envelope. Therefore, an envelope-following filter is used in the algorithm for which the parameters are determined empirically. However, in order to obtain a very smooth estimate of the signal amplitude, a large time constant for the envelope follower should be chosen.

4. TESTING AND RESULTS

The proposed algorithm has been tested using piano sound samples. Figure 7 shows the spectrograms of the original signal and pitch-shifted signals for three different techniques in finding the best splicing point. The signals are of duration 1.0 second and the pitch-shifting factor is equal to $+15\%$ for all the pitch-shifted signals. Figure 7 (a) shows the spectrogram of the original signal, the piano tone B1[1]. In Figure 7 (b), the best splicing point is searched in the original signal using the cross-correlation function, i.e., the standard SOLA method. Figure 7 (c) shows the case in which the best splicing point is searched in the normalized low-pass filtered version of the original signal using the cross-correlation function. In Figure 7 (d), the best splicing point is looked for in the normalized low-pass filtered signal with the AMDF. A linear-phase FIR filter of order 50 with a cut-off frequency of 70 Hz is used for low-pass filtering in both cases. Comparing the spectrogram of the original signal and those of the pitch-shifted signals reveals that the pitch-shifting process brings about discontinuities in the signal which appear as dark vertical lines in the figures (a few examples are circled in Fig. 7). These dark lines are the splicing points whose occurrence may be heard as clicks depending on the signal content at these points.

Figures 7 (b) and (c) show that when the signal is low-pass filtered and normalized before the splicing point search, the number of discontinuities and their intensities are reduced. These changes and improvements in the discontinuities are also audible in the sound samples. In the last case (Figure 7 (d)), where the NFC-TSM method has been used in the pitch-shifting process, no clicks are heard according to our informal listening tests [23]. This is because the continuity of the signal is preserved for low-frequency content of the signal. Moreover, small changes in the spectrogram have occurred, but many of the vertical lines are still visible. On the whole, searching the best splicing point in the normalized low-pass filtered signal leads to a smaller number of discontinuities than when searching in the original signal. The audibility of clicks and masking effects in the auditory system are beyond the scope of this paper, and are left for future work. In addition, the AMDF performed well in an objective evaluation of the quality of each synchronization procedure presented in [18].

Using this algorithm, real-time pitch-shifting of the input audio is possible. In addition, the pitch-shifting factor can be a time-varying function. For pitch-shifting down, there is no limitation on the amount of pitch-shifting factor. For pitch-shifting up, however, the restriction on the pitch-shifting factor is applied in the algorithm by the period length of the signal and the permitted delay. When pitch-shifting up, it is required to have at least one period of the signal in the ring buffer before the output pointer can jump backward on the ring buffer to repeat a sound segment due to the time-scale modification. This implies a delay in the system. For example, if the fundamental frequency of an input signal is equal to 63 Hz, for an allowed delay of 5 ms, the pitch-shifting up factor can have a maximum amount of $+30\%$. It should be noted that this restriction holds only at the onset of the tone, when pitch-shifting up. After having received one period of the input audio, this limitation is eliminated and, then, pitch-shifting factor can have any value.

In the proposed algorithm, the AMDF calculation is computationally the heaviest part. However, the computation of AMDF is required only when the best splicing point should be found. If the maximum period length of the input signal is T samples, the number of abs, subtraction and addition operations required to find the best splicing point is equal to $C(1-C)T^2$. C is the coefficient to define the lengths of the correlation window and search region ($C = 0.375$). Moreover, $(1 - C)T$ comparisons should be performed to find the minimum AMDF. For example, when the minimum fundamental frequency in the frequency range of the input signal is 73 Hz, the lengths of the correlation window and search area are selected to be 230 and 380 samples, respectively, with the

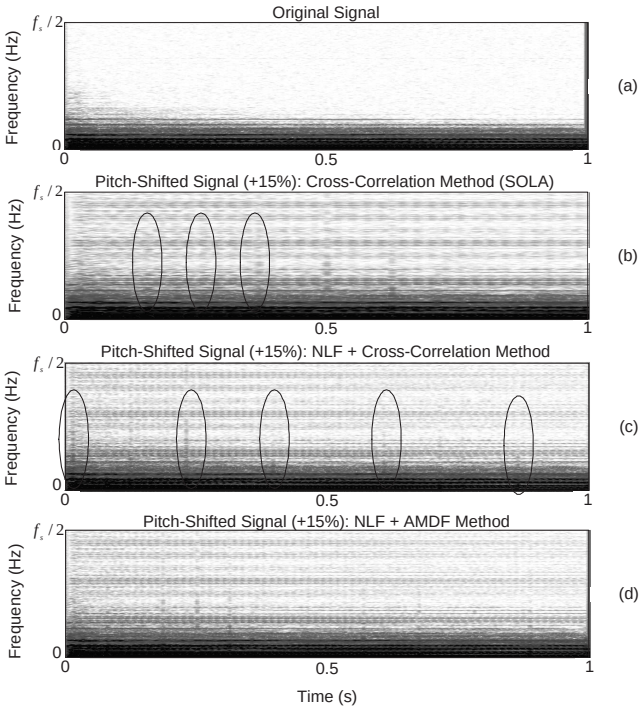


Figure 7: *Spectrograms of (a) the original signal (piano tone B1, fundamental frequency = 63 Hz) and pitch-shifted signals by $+15\%$ using different methods based on (b) search in the original signal using cross-correlation function, (c) search in the normalized low-pass filtered (NLF) version of the original signal using cross-correlation function, (d) search in the normalized low-pass filtered (NLF) version of the original signal using AMDF. The ovals in the figures indicate occurrence of audible artifacts.*

[1]The sound sample has been taken from the McGill University Master Samples Collection.

sampling frequency of 44.1 kHz. Therefore, every time the best splicing point is searched in the ring buffer 87400 abs, subtraction, and addition operations and 380 comparisons are needed.

Since the search is performed in the normalized low-pass filtered version of the input signal, the smoothness of the normalized filtered signal can be exploited to reduce the sample rate and thus the computational load. According to the period length of the input signal, the correlation window and the search region can be down-sampled when computing AMDF. The longer the period length, the greater the down-sampling factor is. For instance, in the above example a down-sampling factor of 7 can be used without any reduction in the quality of the output signal. This way the number of abs, subtraction, and addition operations can be reduced to 1815 and the number of comparisons to 55. Another method to make the implementation of the algorithm possible, regarding the heavy computational complexity of the AMDF, is to divide the computational load between a number of sound samples before the output pointer jumps in the ring buffer.

This algorithm has been implemented on a Motorola (Freescale) DSP56303 for a monophonic sound and it runs in real-time.

5. CONCLUSIONS

A real-time high-quality pitch-shifting algorithm was presented. The pitch-shifting algorithm is based on the resampling and time-scale modification method. A new method for time-scale modification of musical signals was developed which is called the NFC-TSM technique. In this technique, repeating and discarding signal segments are performed in such a way that the pitch-shifted signal has the highest similarity with the original signal in the case of the changes in the signal attributes. The best point to splice the signal segments is searched in the normalized low-pass filtered signal using the AMDF. In this algorithm, the pitch-shifting factor can have large values without any degradation in the quality of the signal. Sound samples are available at [23].

6. ACKNOWLEDGEMENTS

This work has been supported by the Foundation of Finnish Innovations and CIMO. Special thanks go to Oskari Porkka for the implementation of the algorithm on the Motorola DSP56303.

7. REFERENCES

[1] B. G. Crockett, "High quality multi-channel time-scaling and pitch-shifting using auditory scene analysis," in *Proc. 115th Convention of the Audio Engineering Society, New Yourk, USA, Preprint 5948*, October 2003.

[2] R. Mastro and D. Baldacci, "Real time pitch shift algorithm and enhanced tone controls for deejay applications," in *Proc. 106th Convention of the Audio Engineering Society, Munich, Germany, Preprint 4899*, May 1999.

[3] A. Duncan and D. Rossum, "Fundamentals of pitch shifting," in *Proc. 85th Convention of the Audio Engineering Society, Los Angeles, USA, Preprint 2714*, October 1988.

[4] G. J. Lin, S. G. Chen, and T. Wu, "High quality and low complexity pitch modification of acoustic signals," in *Proc. IEEE Intrnational Conference on Acoustics, Speech, and Signal Processing, vol. 5*, May 1995, pp. 2987–90.

[5] U. Zölzer, Ed., *DAFX - Digital Audio Effects*, J. Wiley & Sons, 2002.

[6] J. L. Flanagan and R. M. Golden, "Phase vocoder," *Bell System Technical Journal*, vol. 45, pp. 1493–509, 1966.

[7] J. Laroche, "Improved phase vocoder time-scale modification of audio," *IEEE Transactions on Speech and Audio Processing*, vol. 7, no. 3, pp. 323–32, May 1999.

[8] D. Rossum, "An analysis of pitch shifting algorithms," in *Proc. of the 87th Convention of the Audio Engineering Society, New York, USA, Preprint 2843*, September 1989.

[9] G. Fairbanks, W. L. Everitt, and R. P. Jaeger, "Method for time or frequency compression-expansion of speech," *Transactions of the Institute of Radio Engineers, Professional Group on Audio AU-2*, pp. 7–12, 1954.

[10] F. F. Lee, "Time compression and expansion of speech by the sampling method," *Journal of the Audio Engineering Society*, vol. 20, no. 9, pp. 738–42, 1972.

[11] D. Knox, N. Bailey, and I. Stewart, "A simple hybrid approach to the time-scale modification of speech," *Journal of the Audio Engineering Society*, vol. 53, no. 7, pp. 612–5, 2005.

[12] D. Dorran and R. Lawlor, "An efficient audio time-scale modification algorithm for use in a subband implementation," in *Proc. of the 6th International Conference on DAFX-03, London, UK*, September 2003.

[13] S. Roucos and A. M. Wilgus, "High quality time-scale modification for speech," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing, Florida, USA*, May 1985, pp. 493–6.

[14] E. Moulines, C. Hamon, and F. Charpentier, "An efficient high quality time-scale modification of speech signal," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing, Glasgow, UK*, May 1989, pp. 238–41.

[15] N. Schnell, G. Peeters, S. Lemouton, P. Manoury, and X. Rodet, "Synthesizing a choir in real-time using pitch-synchronous overlap add (PSOLA)," in *Proc. International Computer Music Conference, Berlin, Germany*, 2000, pp. 102–8.

[16] W. Verhelst and M. Roelands, "An overlap-add technique based on waveform similarity (WSOLA) for high quality time-scale modification of speech," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing, Minneapolis, USA*, April 1993, pp. 554–7.

[17] S. Yim and B. I. Pawate, "Computationally efficient algorithm for time-scale modification (GLS-TSM)," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing, Atlanta, USA*, May 1996, pp. 1009–12.

[18] D. Dorran, R. Lawlor, and E. Coyle, "A comparison of time-domain time-scale modification algorithms," in *Proc. 120th Convention of the Audio Engineering Society, Paris, France, Preprint 6674*, May 2006.

[19] R. W. Schafer and L. R. Rabiner, "A digital signal processing approach to interpolation," *Proc. IEEE*, vol. 61, no. 6, pp. 692–702, 1973.

[20] D. Rossum, "Constraint based audio interpolators," in *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, New Paltz, USA*, October 1993, pp. 161–4.

[21] T. Laakso, V. Välimäki, M. Karjalainen, and U. K. Laine, "Splitting the unit delay-tools for fractional delay filter design," *IEEE Signal Processing Magazine*, vol. 13, no. 1, pp. 30–60, 1996.

[22] P. V. Sankar and L. A. Ferrari, "Simple algorithms and architectures for b-spline interpolation," *IEEE Transactions on Pattern Analysis Machine Intelligence PAMI-10*, vol. 10, no. 2, pp. 271–6, 1988.

[23] A. Haghparast, H. Penttinen, and V. Välimäki, "Sound samples," April 2007, http://www.acoustics.hut.fi/publications/papers/dafx07-pitch-shifting.

[24] J. Laroche, "Autocorrelation method for high-quality time/pitch-scaling," in *Proc. IEEE Workshop Application of Signal Processing to Audio and Acoustics, New York, USA*, October 1993, pp. 131–4.

[25] J. B. Hong, "An efficient high quality time-scale modification of speech signal," in *Proc. Finnish Signal Processing Symposium*, May 1997, pp. 139–42.

REAL-TIME REVERB SIMULATION FOR ARBITRARY OBJECT SHAPES

Cynthia Bruyns Maxwell
University of California at Berkeley
Computer Science Department
Center for New Music and Audio Technologies
Berkeley, California USA
cbruyns@cs.berkeley.edu

ABSTRACT

We present a method for simulating reverberation in real-time using arbitrary object shapes. This method is an extension of digital plate reverberation where a dry signal is filtered through a physical model of an object vibrating in response to audio input. Using the modal synthesis method, we can simulate the vibration of many different shapes and materials in real time. Sound samples are available at the follwing website:
http://cynthia.code404.com/dafx-audio/.

1. INTRODUCTION

Historically, plate reverberation was used as a synthetic means to simulate large room acoustics. It was one of the first types of artificial reverberation used in recording [1]. Despite the unnatural sound produced as compared to large room reverberation, plates were used extensively due to their relative low cost and small size. Recently, researchers have looked for a means of digitally simulating plate reverberation to recreate this unique analog recording style [2].

Analog plate reverberation works by mounting a steel plate with tension supplied by springs at the corners where the plate is attached to a stable frame. A signal from a transducer is applied to the plate, causing it to vibrate. This vibration is then sensed elsewhere on the plate with contact microphones. A nearby absorbing pad can also be used to control the near-field radiation.

Before the prevalence of physical modeling, plate reverberation was simulated by recording impulse responses of plates made from different materials and different geometries. By convolution of the input with the impulse response of choice, the resulting audio could sound as though it was recorded through an actual plate. Although this method is very efficient, it is limited by the number of, and variations in, the recordings available.

Bilbao et al. [2] demonstrated a model for plate reverberation using a linear Kirchoff plate formulation. By using a physical model instead of convolution with impulse responses, they could modify the geometry of the plate and input/output parameters. To simulate plate vibration, the model was discretized in space and time using finite differences. One drawback of this method, however, was the large performance requirements preventing their model from running in real-time on an average digital workstation.

Using the modal synthesis method, we can compute a plate reverberation model in real-time and still allow for modifications of the plate and input/output parameters. To achieve this performance, we use the same finite element model as described in [3] and apply forces using the discrete convolution integral method as described in [4] and [5]. We implement this reverberation as an effect plug-in that takes an audio stream as the input and produces the sound of the object vibration as the output.

The main contributions of this paper are to extend the use of the modal synthesis and the discrete convolution integral for the rapid simulation of the motion of objects in response to arbitrary loading profiles. We give examples of using this technique for the deformation of linear shell models of simple and complex shapes in a real-time synthesis environment.

2. METHODS

To briefly review the steps in obtaining a resonator bank from a arbitrary geometry we begin by discretizing the Mindlin/Reissner thin plate equations as found in [6] and [7]. This plate model differs from Kirchoff plate theory by adding the effects of shear deformation across the plate thickness. This additional motion allows for modeling thin and thick plates. We also extend the plate model to include membrane forces essentially creating a shell model from planar elements as described in [8]. The planar shell elements can be made of quadrilateral or triangular patches.

This element allows for five degrees-of-freedom at each vertex as demonstrated in Figure 1. In-plane displacement is captured by the degrees-of-freedom, u and v. For out-of-plane motion, bending is represented by adding the rotational degrees-of-freedom, $\theta_x = \frac{dw}{dx}$ and $\theta_y = \frac{dw}{dy}$, about the x and y axes, as well as an out-of-plane displacement, w. This formulation leads to the following element stiffness representation that accounts for membrane K_m, shear K_s and bending stiffness K_b (Equation 1).

$$k^e = \int_{\Omega_e} B_b^T D_b B_b d\Omega + \int_{\Omega_e} B_s^T D_s B_s d\Omega + \int_{\Omega_e} B_m^T D_m B_m d\Omega \quad (1)$$

where D represents the constitutive matrices for each stress condition, N represents the interpolating functions and B represents the operator applied on these functions. The exact entries in N and B depend on the number of nodes per element and the order of the interpolating polynomials used.

Similarly, the element mass matrix is represented as:

$$m^e = \int_{\Omega_e} N^T N d\Omega \quad (2)$$

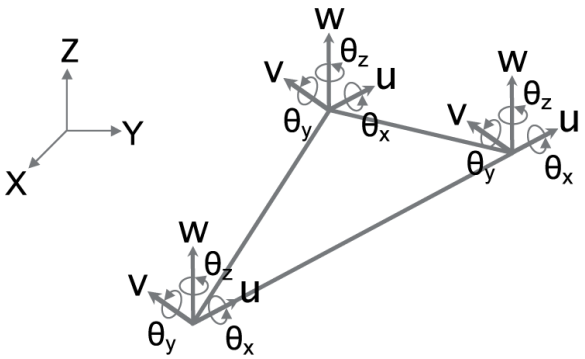


Figure 1: Normal, shear and bending forces and moments.

A more detailed discussion of the exact entries in these matrices is given in [6] and [7].

We sum the element matrices, k^e and m^e together to form an overall algebraic representation of the system. The result is the canonical representation for the second order partial differential equation of motion:

$$M\ddot{u} + D\dot{u} + Ku \quad = \quad F(t) \tag{3}$$

where M is the matrix representing the distribution of mass in the system, D is a measure of damping, K is the stiffness matrix and $F(t)$ is the force applied to the object over time. This equation expresses the balance of forces generated by the acceleration, velocity and displacement of the object. In this form, the system of equations for each degree-of-freedom (DOF) are coupled and thus the solution involves manipulation of these large system matrices. Alternatively, modal analysis seeks to decouple this system into single DOF oscillators. For a detailed description of the modal synthesis method as well as a comparison with other physical modeling techniques see [3] and references therein.

Without damping, the procedure for uncoupling these equations is straightforward using the general eigenvalue decomposition $Kx = \lambda Mx$. However, with damping, decoupling these equations requires some assumptions to be made [9].

In many finite element representations, there is no straightforward method of generating the damping matrix D. "A major reason for this is that, in contrast with inertia and stiffness forces, the physics behind the damping forces is in general not clear. As a consequence, modelling of damping from the first principles is difficult, if not impossible, for real-life engineering structures. The common approach is to use the proportional damping model, where it is assumed that the damping matrix is proportional to mass and stiffness matrices [10]." This proportionality is represented as:

$$D = \alpha_1 M + \alpha_2 K \tag{4}$$

where $\alpha_1$and α_2 are real scalars. This damping model is also known as Rayleigh damping or classical damping. Modes of classically damped systems preserve the simplicity of the real normal modes as in the undamped case.

The main limitation of the proportional damping approximation comes from the fact that the variation of damping factors with respect to vibration frequency cannot be modelled accurately by using this approach [10]. Research into the error introduced by assuming proportional damping is ongoing and current results seem to suggest that there may never be one static assumption that accurately diagonalizes a coupled damped system [11]. For now we use the proportional damping model for its simplicity.

Substituting back into Equation 3, we have:

$$M(\ddot{u} + \alpha_1\dot{u}) + K(\alpha_2\dot{u} + u) \quad = \quad F(t) \tag{5}$$

We assume a particular solution of the form:

$$u \quad = \quad Zv \tag{6}$$

where Z is the matrix of eigenvectors that diagonalizes the system. Substituting back into Equation 3 and pre-multiplying by Z^T we have:

$$Z^T MZ(\ddot{v} + \alpha_1\dot{v}) + Z^T KZ(\alpha_2\dot{v} + v) \quad = \quad Z^T F(t) \tag{7}$$

This equation simplifies to:

$$\ddot{v} + (\alpha_1 + \alpha_2\omega^2)\dot{v} + \omega^2 v \quad = \quad Z^T F(t) \tag{8}$$

Equation 8 represents the uncoupled bank of resonators oscillating at the natural frequencies determined by the eigenvalues of the system. By solving each equation for u we can represent the response at any location on the object at any time. Therefore, to solve for the motion at the pickup locations we weight the contributions of the various modes on the spatial positions of interest.

To apply the input to the system we know that in general, any force-response history can be represented as a succession of infinitesimal impulses. We also know that the response of the system to such a force profile can be built up from the response to each infinitesimal impulse individually [12]. Therefore we can represent a discrete time excitation as a combination of unit step functions:

$$f(n) \quad = \quad \sum_{k=0}^{\infty} f(k)\delta(n-k) \tag{9}$$

where $f(n)$ is the applied force and δ is the Dirac Delta function. The response to this discrete-time excitation is then:

$$x(n) \quad = \quad \sum_{k=0}^{\infty} f(k)g(n-k) = \sum_{k=0}^{n} f(k)g(n-k) \tag{10}$$

where $g(n)$ represents the discrete-time impulse response of a linear time invariant system to the unit impulse $\delta(n)$. Equation 10 essentially approximates the response $x(n)$ in the form of a convolution sum, the discrete counterpart of the convolution integral. The convolution or Duhamel's integral is a means of finding solutions to linear, nonhomogeneous, second order, ordinary differential equations with constant coefficients. The nonhomogeneous part of the equations comes from the forcing function and depending on the complexity of this term, the integral may or may not have a closed-form solution [13]. The convolution integral then, is a means of finding the solution to the original system by summing the individual impulse responses.

Instead of evaluating the non-recursive convolution sum as in Equation 10, DiFilippo and Pai [4] use a different technique to solve for the response to non-harmonic excitation. They use a recursive method for approximating the response at the current time

step by scaling the value at the previous time step and adding that to the response to the new impulse. Thus, for each resonator:

$$x_n(0) = a_n f(0) \quad (11)$$

$$x_n(k) = e^{i\frac{\Omega_n}{F_s}} x_n(k-1) + a_n f(k) \quad (12)$$

where Ω_n is the natural frequency and F_s is the sampling frequency and:

$$\Omega_n = \omega_n + i d_n \quad (13)$$

$$d_n = \frac{\omega_n}{2\pi}\pi \tan(\phi) \quad (14)$$

where ϕ is an internal damping factor. Thus the overall sound generated at time t is:

$$s(t) = Re\left(\sum x_n(k)\right) \quad (15)$$

In this way, the incoming audio signal represents an arbitrary discretized force profile applied to the resonators interpolating the input position.

2.1. Software Implementation

The rendering algorithm works by first performing the modal decomposition and then filtering the incoming audio through the resonator bank produced. The time to compute the modal decomposition depends on the number of modes required and the number of elements in the finite element model. We achieve real-time performance by first computing the decomposition, which can take several seconds. We then evaluate Equation 15 for each audio sample. $s(t)$ is computed in a parallel (or vectorized) fashion as each mode is linearly independent.

The user interface for the plug-in loads an object geometry and displays the surface for specifying the input and pickup locations. The left portion of the user-interface allows for modification of the material parameters, object scale and plate thickness. These parameters are adjusted before modal decomposition. The right portion of the user interface has controls for the audio rendering parameters such as the frequency scaling and resonator decay. These parameters do not require reanalysis, instead they are applied to the bank of resonators as audio is rendered. There is also control for the number of resonators used for simulation. Using more resonators creates a fuller tone but requires more computation.

3. RESULTS

The following examples were computed using one processor of a dual 2.5GHz Power PC G5. The plug-ins were hosted using Apple Inc.'s AULab application and audio input was streamed using the built-in AUFilePlayer component. In each example, the points in green represent the input position and the points in red represent the pickup locations.

For the first example, we load a simple plate model as shown in Figure 2 (top). The model has 100 elements, and the time to compute the decomposition into 485 modes was 0.65 seconds. Figure 3 (top) shows the waveform and Figure 4 (top) shows the spectrogram of the incoming signal applied to the plate. Figure 3 (middle) shows the resulting waveform and Figure 4 (middle) shows the frequency profile generated for the left channel. Immediately, one can see the effect of reverberation on the resulting audio. Where there were once discrete peaks, the audio now blends together. Moreover, the frequency spectrum is low-pass filtered through the number of modes used in the synthesis algorithm.

We can use also this method on novel shapes and explore the effect on the resulting audio. Figure 2 (bottom) shows a more complex shell surface with arbitrary input and output locations. This model had 500 elements and took 24.5 seconds to compute all 1548 modes. Using the same input profile as Figure 3 (top), we can compare the resulting waveform and frequency spectra when rendering through this new geometry (Figure 3 (bottom), Figure 4 (bottom)).

Notice that in Figure 4, the output through the resonator bank has less of the high frequency components than the original signal. This is to be expected as the resonant frequencies of the set of resonators and user-selected damping values will not exactly match the original signal. In some sense, the original signal acts to imprint its frequency spectrum on the resonator bank roughly but need not exactly match.

For both of these examples, simulating object vibration using 20 modes consumed around 1.4% CPU; 100 modes consumed roughly 3%; 1000 modes consumed 22%; and 3000 modes used 84% for two channels of stereo processing. These results suggest that for up to 1000 modes, the method performs well. For the 3000 or so resonators needed for non-metalic, perceptually realistic sounding reverberation [14], the real-time CPU demand is considerable when using only one processor.

4. DISCUSSION

In this investigation we have demonstrated a method for simulating reverberation using the modal synthesis method. By using a physical model of a vibrating object, we are free to use any arbitrary geometry and material.

For plates with very thin cross-sections, it is likely that large applied forces will cause large plate deformation. When this happens, linear models can no longer be used. As a result, techniques such as linear modal superposition must be abandoned for nonlinear modal analysis or nonlinear models and numerical integration. Other researchers are actively investigating the importance of these nonlinearities in plate reverberation models [15].

5. ACKNOWLEDGEMENTS

Thanks to John Lazzaro, Antoine Chaigne and Julius O. Smith III for useful discussions on plate reverberators and to Justin Maxwell for the user interface layout. Also thanks to C.J. Slyfield for editing the complex shape model.

6. REFERENCES

[1] D. C. Myers, "Audio reverberator," Tech. Rep., U.S. Patent 4,653,101, March 24, 1984.

[2] A. Bilbao, K. Arcas, and A. Chaigne, "A physical model for plate reverberation," *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, vol. 5, pp. V–165–V–168, 2006.

[3] C. Bruyns, "Modal synthesis for arbitrarily shaped objects," *Computer Music Journal*, vol. 30, no. 3, pp. 22–37, 2006.

[4] D. DiFilippo and D.K. Pai, "The AHI: An audio haptic interface for contact interactions," in *Proceedings of ACM UIST 2000*, 2000.

[5] K. van den Doel, P. G. Kry, and D.K. Pai, "Physically-based sound effects for interactive simulation and animation," in *Proceedings of SIGGRAPH 2001*, 2001.

[6] S. S. Rao, *The Finite Element Method in Engineering Second Edition*, Pergamon Press, Oxford, 1989.

[7] Y. W. Kwon and H. Bang, *The finite element method using MATLAB*, CRC Press, 2000.

[8] O. C. Zienkiewicz and R. L. Taylor, *Finite Element Method: Volume 2, Solid Mechanics*, Butterworth-Heinemann, Burlington, MA, 5th edition, 2000.

[9] G. B. Warburton and S. R. Soni, "Errors in response calculations for non-classically damped structures," *Earthquake Engineering & Structural Dynamics*, vol. 5, no. 4, pp. 365 – 376, 1977.

[10] S. Adhikari, "Damping modelling and identification using generalized proportional damping," in *Proceedings of the 23rd International Modal Analysis Conference (IMAC-XXIII), Orlando, Florida, USA*, Feb 2005.

[11] F. Ma and T.K. Caughey, "Analysis of linear nonconservative vibrations," *Journal of Applied Mechanics*, vol. 62, pp. 685–691, 1995.

[12] L. Meirovitch, *Principles and Techniques of Vibrations*, Prentice Hall, Upper Saddle River, New Jersey, first edition, 1999.

[13] E. Chicurel-Uziel, "Closed-form solution for response of linear systems subjected to periodic non-harmonic excitation," in *Proceedings of the Institute of Mechanical Engineers, Part K*, 2000.

[14] M. Karjalainen and H. Järveläinen, "More about this reverberation science: Perceptually good late reverberation," in *In Proceedings of the AES 111th International Convention*, September 2001.

[15] A. Chaigne, "Plate reverberator: modeling, analysis and synthesis," CCRMA Music 420 Seminar, March 2007.

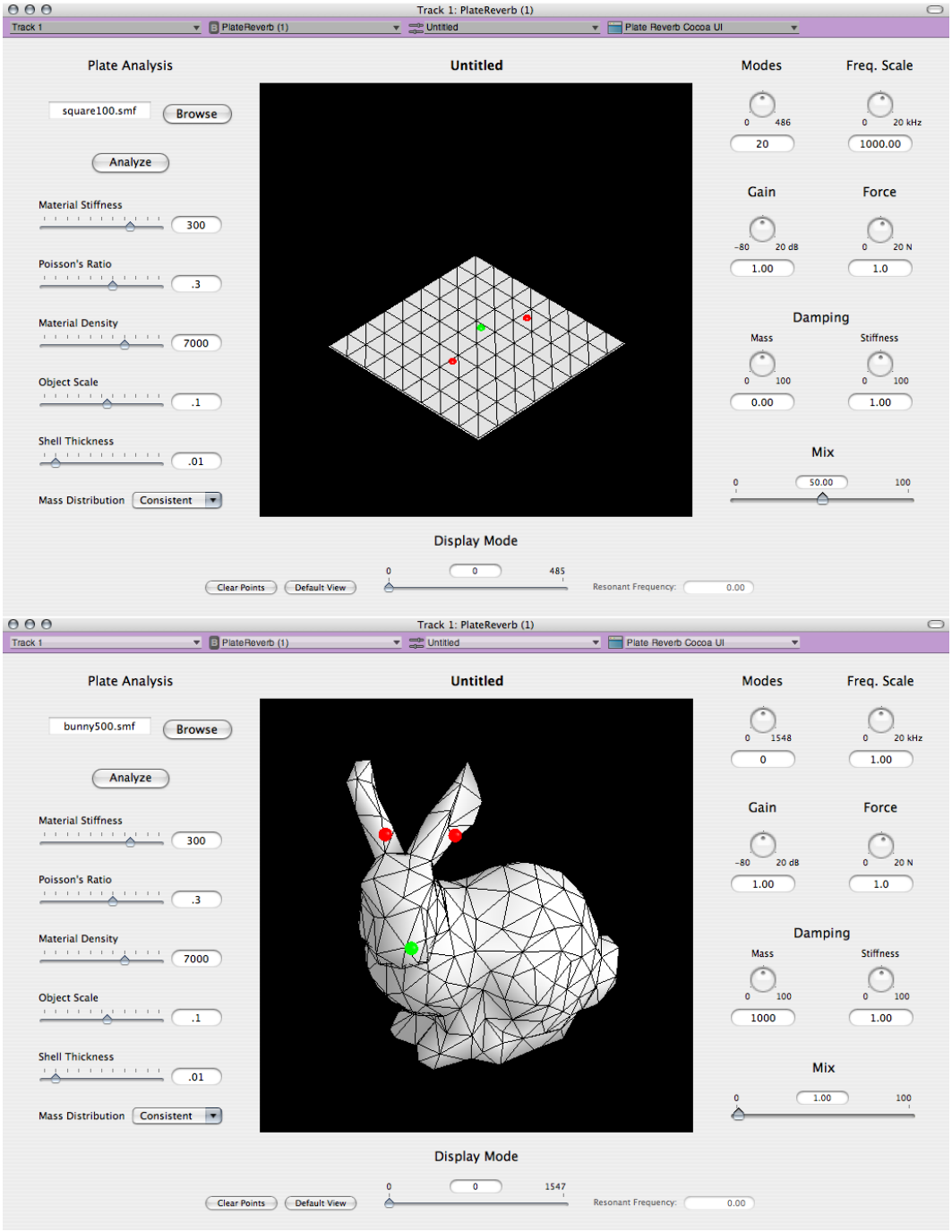


Figure 2: Top: A traditional plate reverb geometry. Bottom: A complex reverb surface.

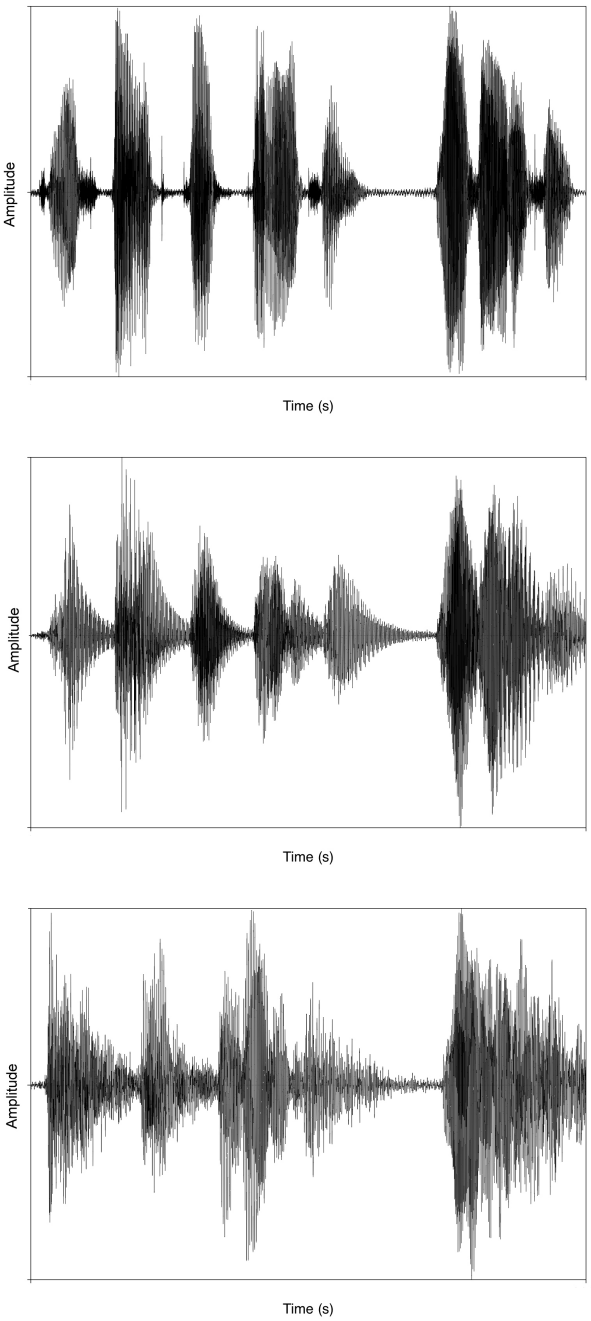


Figure 3: Top: Incoming audio signal applied to the plate. Middle: Simple plate vibration. Bottom: Complex surface vibration.

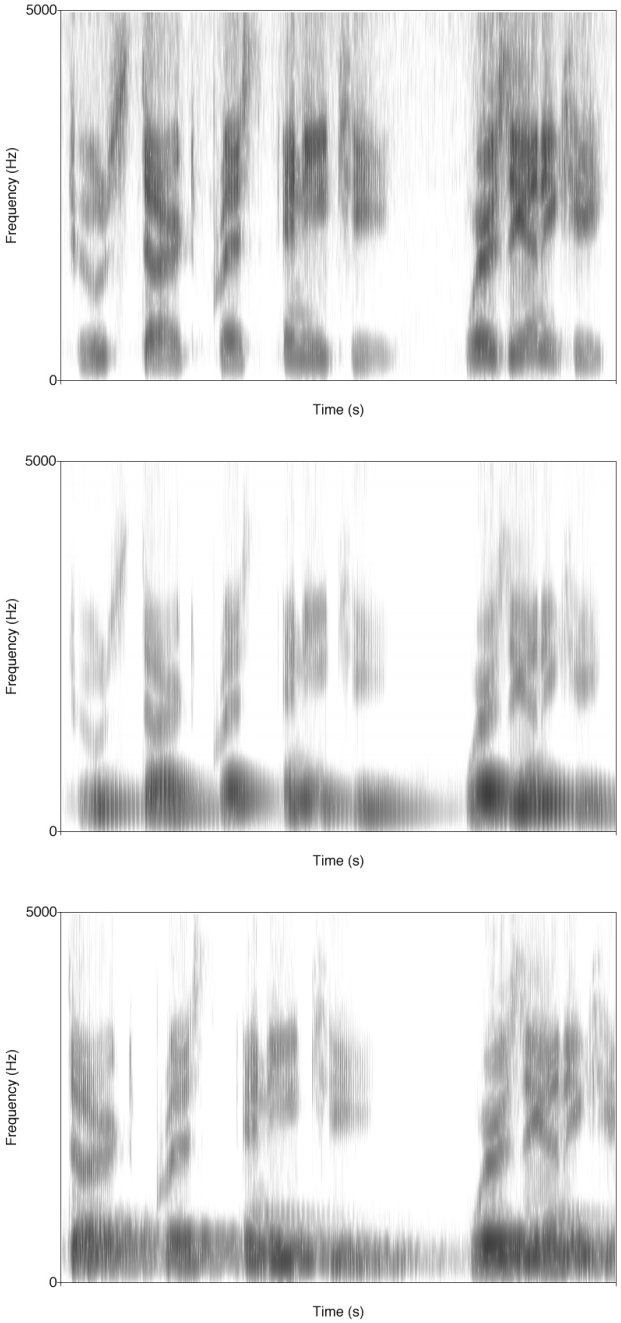


Figure 4: Top: Frequency profile of the force profile. Middle: Simple plate vibration. Bottom: Complex surface vibration.

ADAPTIVE FM SYNTHESIS

Victor Lazzarini, Joseph Timoney and Thomas Lysaght

Sound and Music Technology Research Group
National University of Ireland, Maynooth
Victor.Lazzarini@nuim.ie, {JTimoney,TLysaght}@cs.nuim.ie

ABSTRACT

This article describes an adaptive synthesis technique based on frequency (phase) modulation of arbitrary input signals. The background and motivation for the development of the technique, as well as related work, are discussed. A detailed description of delay line-based phase modulation of sinusoidal and complex signals is provided. The basic design of an implementation of the technique is presented and commented. A series of examples using four different instrumental sources are discussed. The results show a wide range of possible effects through the use of the technique, from addition of higher components, to changes in the odd-even harmonic balance and the introduction of controlled inharmonicity.

1. INTRODUCTION

Adaptive digital audio effects[1] form an important subset of musical signal processing techniques. A key aspect of their usefulness in music composition and performance is that they provide a means to retain significant gestural information contained in the original signal. One of the major criticisms of electronic and computer music is the relative lack of gestural control over the sonic result. With the use of adaptive techniques, it is possible to re-inject much of the liveliness perceived in musical signals of instrumental origin.

Frequency modulation (FM) synthesis[2] is classic synthesis method which has been very useful as an economic means to generate time-varying complex spectra. However, one of its limitation s, as is the case with many of the classic techniques, has always been the difficulty in producing more natural-sounding spectral evolutions, due to the lack of fine gestural control over the sound. The usual methods of controlling FM synthesizers, such as keyboards, modulation-wheels, sliders and breath controllers never provided more than a basic means of dynamically modifying the synthesized signal. The limitation might be related to the lack of resolution of these gestural controllers, but it is also related to the paradigm of control employed.

The traditional approach has been to treat synthesis and control parameters separately, which ultimately will lead to a split between gesture and sound result. An alternative is provided by approaching the problem from the adaptive point of view, whereby the synthesis parameters are derived from the input signal itself. In addition, in our proposed method, the input signal itself is used in the synthesis process. By extracting significant information from the input signal and applying it as a way of controlling the modulation of the same signal, we have arrived at a gesture-aware form of FM synthesis, Adaptive-FM (AdFM).

1.1. Related work

The technique described here lies in the area defined by Poepel and Dannenberg[3] as 'audio-signal-driven' sound synthesis. These authors have proposed a number of correlate techniques, including a FM synthesis method using instrumental input as modulation sources and a variation on the technique proposed here, which they named 'self-modulation'.

Their FM synthesis approach differs from AdFM in that it is a case is of single-carrier complex modulation, whereas here we propose a technique similar to multi-carrier FM. As hinted above, the method of self-modulation uses a similar basic signal processing principle. However, it differs substantially from our approach in very significant points (which will generally result in the lack of fine control over the process). These differences will be discussed below.

More traditionally, some classic signal processing algorithm designs also lie in the same general area. Different methods of waveshaping[4][5] of arbitrary input signals, as well as single-side band[6] and ring modulation are very much related to the present work. However, none of these provide as fine control over the resulting synthetic output as ADFM.

2. THE TECHNIQUE OF ADFM

The synthesis method provided here is based on two elements: the employment of a variable delay line as a means of phase modulation of a signal and the use of an arbitrary input, to which parameter estimation will be applied.

2.1. Delay line-based phase modulation

A well-known side-effect of variable delays is the phase modulation of the delay line input. This is the basis for all classic variable-delay effects such as flanging, chorus, pitch shifting and vibrato. It is thus possible to model simple (sinusoidal) audio-rate phase modulation using a delay-line with a suitable modulating function (Fig.1).

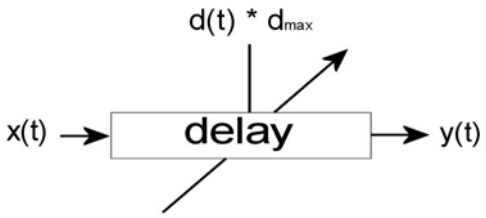


Figure 1. *Delay-line phase modulation*

We now consider the case where the input to the delay line is a sinusoidal signal of frequency f_c :

$$x(t) = \sin(2\pi f_c t) \tag{1}$$

The instantaneous frequency *IF(t)* of the phase-modulated signal is given by the following relationship [7]:

$$IF(t) = -\frac{\partial d(t)}{\partial t} d_{\max} f_c + f_c \tag{2}$$

where *d(t)* is the modulating signal and d_{max} is the maximum delay in seconds. Using a scaled raised cosine as a modulating function

$$d(t) = 0.5\cos(2\pi f_m t) + 0.5 \tag{3}$$

we have (by substituting *d(t)* in Eq.2)

$$IF(t) = \pi f_m \sin(2\pi f_m t) d_{\max} f_c + f_c \tag{4}$$

which characterises sinusoidal frequency (phase) modulation. In such arrangement, the sinusoidal term in Eq(4) is known as the frequency deviation, whose maximum absolute value DEV_{max} is:

$$DEV_{\max} = \frac{\Delta d \pi}{T_m} f_c = \Delta d \times \pi f_m f_c \tag{5}$$

with $\Delta d = d_{max} - d_{min..}$

Now, turning to FM theory, we characterise the index of modulation I as the ratio of the maximum deviation and the modulation frequency:

$$I = \frac{DEV_{\max}}{f_m} = \frac{\Delta d \pi f_m f_c}{f_m} = \Delta d \pi f_c \tag{6}$$

So, the Δd that should apply as the amplitude of our sinusoidal modulating signal can now be put in terms of the index of modulation

$$\Delta d = \frac{I}{\pi f_c} \tag{7}$$

and the modulating signal is now:

$$d(t) = \frac{I}{\pi f_c}\left[0.5\cos(2\pi f_m t) + 0.5\right] \tag{8}$$

The resulting spectra according to FM theory is dependent on the values of both I and the *fc:fm* ratio:

$$y(t) = J_0(I)\sin(\omega_c t) + \sum_{k=1}^{I+1} J_k(I)\sin(\omega_c t + k\omega_m t) + J_{-k}(I)\sin(\omega_c t - k\omega_m t) \tag{9}$$

where $\omega_c = 2\pi f_c, \omega_m = 2\pi f_m, J_k(I)$ are Bessel functions of the 1st kind of order k and

$$J_{-k}(I) = (-1)^k J_k(I) \tag{10}$$

Interestingly enough, in the delay-line formulation of FM/PM, the index of modulation for a given variable delay width is proportional to the carrier signal frequency. This situation does not arise in classic FM. Also, when considering the width of variable delay for a given value of I, we see that it gets smaller as the frequency rises. In a digital system, for I=1, this will be less than 1 sample at the Nyquist frequency.

2.2. Using an arbitrary input signal

In Eq.9, we see the ordinary spectrum of simple FM. However, for our present purposes, we will assume the input to be a complex arbitrary signal made up of N+1 sinusoidal partials of amplitudes a_n , radian frequencies ω_n and phase offsets ϕ_n ,originating, for instance, from instrumental sources:

$$x(t) = \sum_{n=0}^{N} a_n \sin(\omega_n t + \phi_n) \tag{11}$$

The resulting phase-modulated output is equivalent what is normally called multi-carrier FM synthesis, since the carrier signal is now complex. This output can be described as

$$y(t) = \sum_{n=0}^{N} a_n \sin(\omega_n t + I_n \sin(\omega_m t) + \phi_n) \tag{12}$$

where ω_m is the modulation frequency and I_n is the index of modulation for each partial. According to Eq.9, this would be equivalent to the following signal:

$$y(t) = \sum_{n=0}^{N} a_n \left[\begin{array}{c} J_0(I_n)\sin(\omega_n t + \phi_n) + \\ \sum_{k=1}^{I+1} \left\langle J_k(I_n)\sin(\omega_n t + k\omega_m t + \phi_n) + \right. \\ \left. J_{-k}(I_n)\sin(\omega_n t - k\omega_m t + \phi_n) \right\rangle \end{array} \right] \tag{13}$$

The different indexes of modulation for each component of the carrier signal can be estimated by the following relationship:

$$I_n = \Delta d \pi f_n = \frac{I}{\pi f_o} \pi f_n = I \frac{f_n}{f_o} \tag{14}$$

Again, we see here that the effect of the relationship between the index of modulation and the carrier frequency is that higher partials will be modulated more intensely than lower ones. Depending on the bandwidth and richness of the input signal, it is quite easy to generate very complex spectra, which might be objectionable in some cases.

This is indeed the case in the related technique of self-modulation, where both the modulating signal and the carrier are complex. However, since here we have full control of the index of modulation and we have a sinusoidal modulator, it is possible to realise more subtle and controlled FM.

Another key aspect of the proposed method is that the *fc:fm* ratio parameter can be also be taken advantage of by estimating the fundamental frequency of the input signal (assumed to be monophonic). In this case, a variety of different spectral combinations can be produced, from inharmonic to harmonic and quasi-harmonic.

2.3. Input signal parameter estimation

In order to allow for a full control of *fc:fm* ratio and modulation index, it is necessary to estimate the fundamental frequency of the carrier signal. That will allow the modulator signal frequency and amplitude to be set according to eq.8. This can be achieved with the use of a pitch tracker, which is a standard component of any modern musical signal processing system. For the current implementation, a spectral analysis pitch tracking method was devised, based on an algorithm by M Puckette et al [8][9], which provides fine accuracy of fundamental frequency estimation. In addition to tracking the pitch, it is also useful to obtain the amplitude of the input signal, which can be used in certain applications to scale the index of modulation. This is also provided by our parameter estimation method.

2.4. Bandwidth and aliasing issues

Although the spectrum of FM is band-limited, it is capable of producing very high frequencies, according to eqs.9 and 13. With digital signals this can lead to aliasing problems, if the bandwidth of the signal exceeds the Nyquist frequency. The fact that the index of modulation increases with frequency, for a given Δd, as seen in eq.14, is obviously problematic. However, in practice, since the kind of input signals we will be employing generally exhibit a spectral envelope that decays with frequency, objectionable aliasing problems might be greatly minimised, given that a_n, in eq.13, for higher values of n will be close to 0. Of course, if our input contains a lot of energy in the higher end of the spectrum, such as for instance an impulse train, then aliasing will surely occur.

The simplest solution for such problematic signals is to impose a decaying spectral envelope by the use of a filter. This will have the obvious side-effect of modifying the timbre of the input signal. Another, more computationally costly, solution is to oversample the input signal. This would either remove the aliased signals or place them at an inaudible range.

3. IMPLEMENTATION

The basic design of AdFM is shown on fig.2. There are three basic components: i) pitch tracker ii) modulating source (a table-lookup oscillator) and iii) variable delay line with interpolated readout. All of these components are found on modern musical signal processing systems, so the technique is highly portable. The implementation discussed here uses the Python[10] language (for control) and Csound 5[11] (as the synthesis engine). It is important to note that this design can be used either for realtime or 'off-line' applications. In addition, plugins can be easily developed from it using csLadspa [12] .

The equivalent Csound 5 code for the design on fig.2 is shown below, with comments:

```
/* AdFM opcode

  asig AdFM  ain,krat,kndx,ifn

  ain – input signal
  krat – fc:fm ratio
  kndx – index of modulation

    ifn – mod signal function table
*/
opcode AdFM,a,akkiii
 setksmps 1
 ipi = 3.1415926
 ioff = 2/44100 /* 2-sample offset*/
 as,krt,knx,ilo,ihi,ifn xin
 /* pitch tracking */
 kcps,kamp ptrack as,512
 /* modulator */
 adt oscili knx/(ipi*kcps),kcps/krt,ifn
 adp   delayr 1
 /* delay line */
 adel  deltap3  adt + ioff
       delayw  as
       xout adel
endop
```

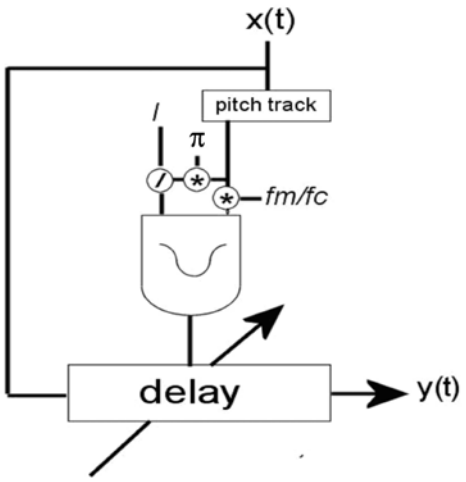


Figure 2. *The Basic AdFM design*

This implementation uses a spectral analysis pitch tracking opcode written by the authors, a linear interpolation oscillator to generate the modulation signal and a cubic interpolation variable delay line. Due to the use of cubic interpolation, the minimum delay is set to 2 samples to avoid errors in the circular buffer readout.

A number of variations can be made to the basic design. For instance, the amplitude of the signal, which is produced together with pitch tracking, can be used to scale the index of modulation. This will generate so called brass-like tones[13], where the brightness of the synthetic output will be linked to the amplitude evolution of the input sound. Alternatively, it can be used to determine the *fc:fm* ratio.

Depending on the characteristics of the input signal, it might be useful to include a lowpass filter before the signal is sent to the AdFM processor. The cutoff frequency of the lowpass filter can also be controlled by the estimated input amplitude. As discussed

earlier, this will reduce aliasing as well as overall brightness, which sometimes is a downside of FM synthesis.

The basic design and its variations have been combined in a computer instrument written in Python, using the Csound 5 API, with a GTK-based graphic user interface (using PyGTK). All the synthesis parameters are exposed by the GUI, so the user can adjust the technique to suit his/her needs.

4. EXAMPLES AND DISCUSSION

Four different types of carrier signal were chosen as a way of examining the qualities of the AdFM synthetic signal. A flute input with its spectral energy concentrated in the lower harmonics was a prime candidate for experimentation. The clarinet was chosen for its basic quality of having more prominent odd harmonics. Finally, the piano and voice were used as a means of exploring the possibilities of synthesising different types of harmonic and inharmonic spectra by the use of various f_c:f_m ratios.

4.1. Flute input

The original flute spectrum (steady-state), effectively with I=0, is shown on fig.3. As clearly seen in that figure, it features quite prominent lower harmonics. By applying an index of modulation of 0.3, on a 1:1 f_c:f_m arrangement, we can start enriching the spectrum with higher harmonics (Fig.4). At these low values of I, there is already a considerable addition of components between 5 and 10 KHz. The overall spectral envelope still preserves its original, decaying, shape.

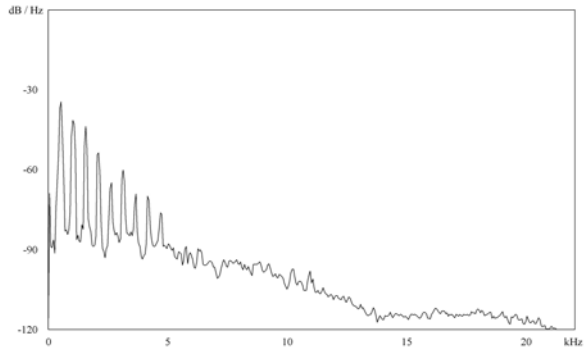


Figure 3. *Steady-state spectrum of a flute playing C4*

With higher values of I, we can see a dramatic change in the timbral characteristics of the original flute sound. Fig.4 shows the resulting spectrum, now with I=1.5. Here, we can see that components are now spread to the entire frequency range. The original decaying spectral envelope is distorted into a much more gradual shape and the difference between the loudest and the softest harmonic is only of about 20 dB. The resulting sound has been described as 'string-like' and the transition between the flute and AdFM spectra is capable of providing interesting possibilities for musical expression. Also, it is important to note that important gestural characteristics of the original sound, such as pitch fluctuations/vibrato and articulation are preserved in the synthetic output.

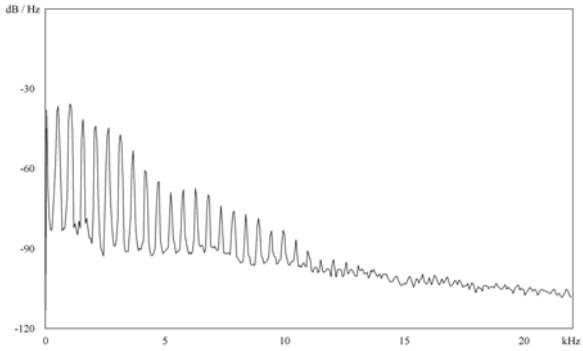


Figure 4. *AdFM spectrum using a flute C4 signal as carrier with fc:fm=1 and I=0.3*

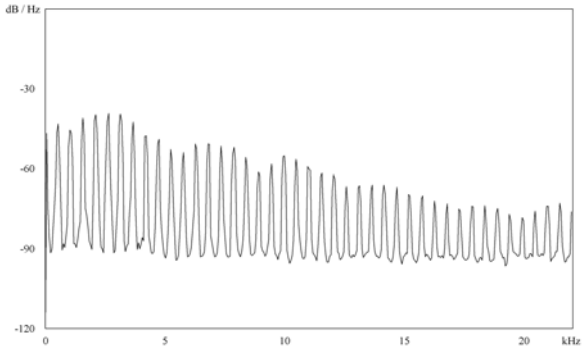


Figure 5. *AdFM spectrum using same input as fig.3, but now with I=1.5*

As I gets higher, the spectrum gets even brighter, but the problems with aliasing start to become significant. To prevent this and also to allow for a different spectral envelope, an optional lowpass filtering of the input signal is suggested. In that case, the filter is inserted in the signal path at the delay-line input. A butterworth low-pass filter with a cutoff frequency between 1000 and 5000 Hz has proven useful. It is possible to couple the cutoff frequency with I, so that for higher values of that parameter, more filtering is applied.

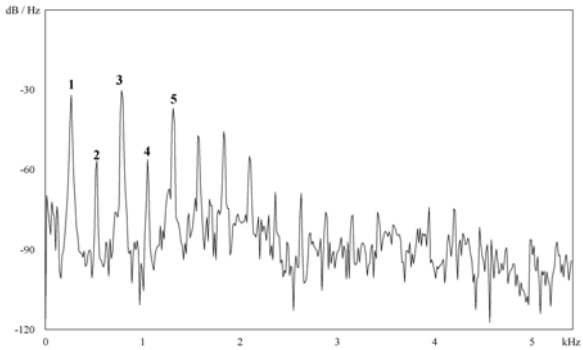


Figure 6. *Detail of steady-state spectrum of clarinet C3. The higher relative strength of lower-order odd harmonics against even ones is clearly seen.*

4.2. Clarinet input

Our second experiment used a clarinet signal as a carrier wave for AdFM. The clarinet exhibits a steady-state spectrum where the lower-order even harmonics are significantly less energetic than its odd neighbours (fig.6). Due to this fact, the multi-carrier-like characteristic of AdFM helps generate quite a change in the spectra of that instrument.

As the index of modulation increases, the balance between odd an even harmonics changes substantially. With I=1.5, it is possible to see that there is now very little difference between the strengths of odd and even components (fig.7). In addition, higher-order harmonics become more present, and the spectral envelope levels out. This is due to the well-know spread of energy that is characteristic of FM synthesis.

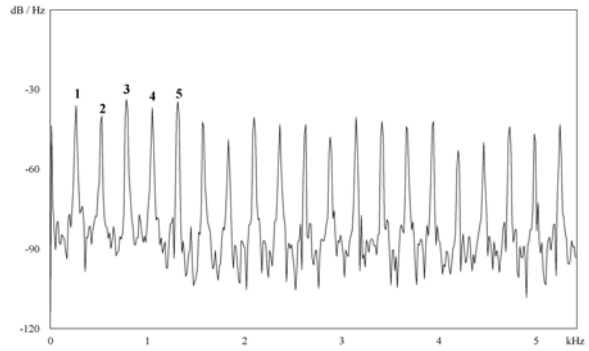


Figure 7**.** *Detail of AdFM spectrum using a clarinet C3 signal as carrier with fc:fm=1 and I=1.5. Odd and even harmonics have now comparable strengths.*

4.3. Piano input

In the previous examples, we have kept the ratio between the modulating frequency and carrier fundamental at 1. However, as we know from FM theory, a range of different spectra are possible if we use different ratios. It is possible to create a range of effects that range from changing the fundamental of the sound to transforming a harmonic spectrum into a inharmonic one. We proceeded to take a piano C2 signal as our carrier and then tuned our modulator to 1.41 times that frequency. The original piano spectrum is shown on fig.8, where we can clearly see its harmonics.

The resulting AdFM spectrum, with I=0.15 is shown on fig.8. This particular ratio creates a great number of components, whose relationship will imply a very low fundamental, thus generating what is perceived as an inharmonic spectrum. With the 1:1 ratio, the sums and differences between f_c and f_m created components whose frequencies were mostly coincident. Here, a variety of discrete components will be generated, creating the denser spectrum seen in fig.9.. The AdFM sound resulting from this arrangement has been described as 'bell-like'. Transitions between piano and bell sounds can be effected by changing I from 0 to the desired value. The application of a lowpass filter at the delay-line input will also allow for some variety and control over the brightness of the result.

4.4. Voice input

A vocal input was used as the fourth different source examined in this work, demonstrating a pitch shift effect. Setting the f_c : f_m ratio to 2, we are able to obtain a sound that is now ½ the pitch of the original. This is due to the introduction of a component at ½ the fundamental frequency corresponding to $f_c - f_m$ in eq.14.

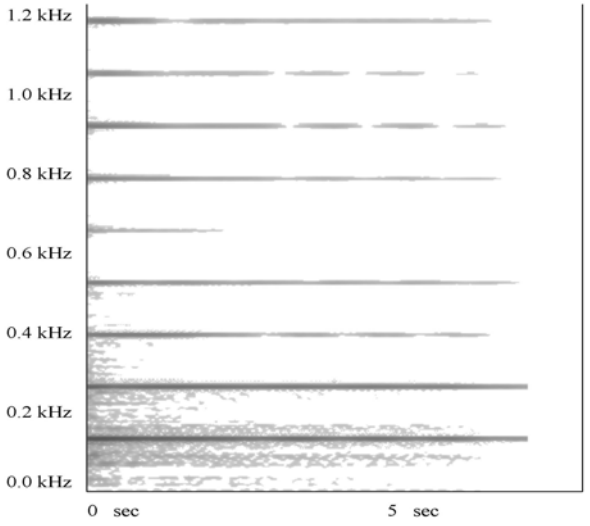


Figure 8. *Spectrogram of a piano C2 tone, showing its first harmonics in the 0-1.2Khz range*

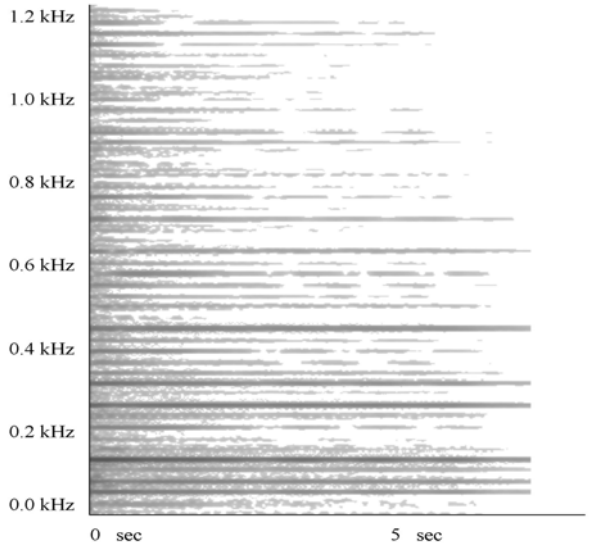


Figure 9. *Spectrogram of an AdFM sound using a piano C2 signal as carrier, with fc:fm=1:1.41 and I=0.5, showing the 0 -1.2Khz range. The resulting inharmonic spectrum, with a large number of components, is clearly seen in comparison with fig. 7*

With the index of modulation at low values (around 0.15), it is possible to preserve some of the spectral shape of the original sound, a crucial step in keeping the intelligibility of the vocal phonemes. Although there is some addition of high frequency components and a flattening of spectral peaks, the AdFM is still perfectly intelligible.

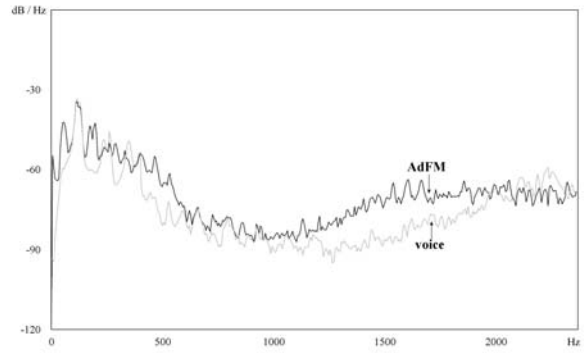


Figure 10. *Comparison of spectral snapshots of a vocal and an AdFM vocal sounds, with I=0.1 and fc:fm=2*

Fig.10 shows a comparison between a vowel steady-state spectrum and its AdFM-processed counterpart. The sub-harmonic peak can be seen at the left of the picture below the original fundamental (a peak at 0Hz is also present, due to the $f_c - 2f_m$ component). The recording of the phrase "this is AdFM Synthesis" is shown as a spectrogram on fig.11., both as the original signal (right) and the AdFM output (left), using the same parameters as above. Again, the octave change is clearly seen, as well as the increase in the number of significant components in the signal.

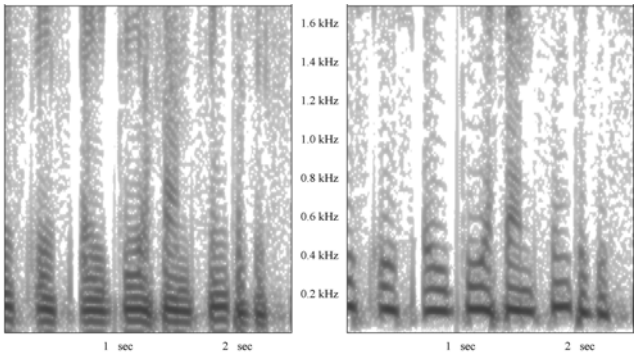


Figure 11. *Detail of spectrogram of a recording of the phrase "this is AdFM synthesis", with AdFM vocal on the left and the original vocal sound on the right*

5. CONCLUSION

We have presented here an alternative approach to the classic technique of FM synthesis, based on an adaptive design. This method belongs to a general class of processes that have been called audio signal driven sound synthesis. Since the FM synthesis theory is well known, it was possible to adapt it to provide a good understanding of the output signal. With the present technique it is possible to have a fine control over the synthetic result, which also preserve a substantial amount of the gestural information in the original signal. Four different types of carrier signal were used in this work to demonstrate the wide range of spectra that the technique is capable of. We are confident this is a simple yet effective way of creating hybrid natural-synthetic sounds for musical applications.

7. REFERENCES

[1] V. Verfaille and D. Arfib, "Implementation Strategies for Adaptive Digital Effects", *Proc. of DAFx02*, pp. 21-26, 2003.

[2] J. Chowning, "The Synthesis of Complex Audio Spectra by Means of Frequency Modulation," Journal of the Audio Engineering Society, 21, pp. 526-34, 1973.

[3] G. Poepel and R. Dannenberg, "Audio Signal Driven Sound Synthesis", *Proc. of ICMC 05,* pp. 391 – 394, 2005.

[4] M. Le Brun, "Digital Waveshaping Synthesis". Journal of the Audio Engineering Society, 27(4), pp.250-266, 1979.

[5] D. Arfib, "Digital synthesis of complex spectra by means of multiplication of non-linear distorted sine waves" , Journal of the Audio Engineering Society, 27(10), pp.757-779, 1979..

[6] S. Wardle, "A Hilbert-Transformer Frequency Shifter for Audio", *Proc. of DAFx98,* pp.25-29, 1998.

[7] S. Dilsch and U. Zoelzer, "Modulation And Delay Line Based Digital Audio Effects", *Proc. of DAFx99*, 2004

[8] M. Puckette and J. Brown, "Accuracy of frequency estimates from the phase vocoder". *IEEE Transactions on Speech and Audio Processing* 6 (2), pp.116-172, 1998

[9] M. Puckette, T. Appel and D. Ziccarelli. *Real-time Audio Analysis Tools for PD and MSP*. *Proceeding s of the ICMC98*, pp. 109-112, 1998.

[10] G. Van Rossum and F. Drake, *The Python Language Reference Manual*, Network Theory, Bristol, 2003.

[11] J. ffitch, "On the Design of Csound5", *Proceedings of the 3rd Linux Audio Conference*, ZKM, Karlsruhe, Germany, pp. 37-42, 2005.

[12] V. Lazzarini and R. Walsh, "Developing LADSPA plugins with Csound", *Proceedings of the 5th Int. Linux Audio Conf.*, TU Berlin, Germany, 30-36, 2007.

[13] J. C. Risset, *An Introductory Catalogue of Computer Synthesized Sounds*, Bell Laboratories, 1969.

ON THE APPLICATION OF RLS ADAPTIVE FILTERING FOR VOICE PITCH MODIFICATION

Rafael C. D. de Paiva, Luiz W. P. Biscainho and Sergio L. Netto

PEE/COPPE, LPS-DEL/Poli
Federal University of Rio de Janeiro
POBox 68504, Rio de Janeiro, RJ, Brazil, 21945-972
(rcdpaiva, wagner, sergioln)@lps.ufrj.br

ABSTRACT

This paper presents a pitch modification scheme, based on the recursive least-squares (RLS) adaptive algorithm, for speech and singing voice signals. The RLS filter is used to determine the linear prediction (LP) model on a sample-by-sample framework, as opposed to the LP-coding (LPC) method, which operates on a block basis. Therefore, an RLS-based approach is able to preserve the natural subtle variations on the vocal tract model, avoiding discontinuities in the synthesized signal and the inherent frame-delay associated to classic methods. The LP residual is modified in the synthesis stage in order to generate the output signal. Listening tests verify the overall quality of the synthesized signal using the RLS approach, indicating that this technique is suitable for real-time applications.

1. INTRODUCTION

Voice analysis and synthesis have been vastly studied in recent years, and many applications and methods have been developed in these areas. Pitch modification, the subject of the present paper, is closely related to voice synthesis, since both systems must consider particular aspects of the voice production system. Applications of voice pitch modification include, for instance, prosody changing, automatic tuning of singing voice and solo to unison transformation. Since it leads to an efficient parameterization of the speech signal, an analysis-and-synthesis scheme for pitch shifting algorithms may also be useful in other applications as concatenative synthesis of voice [1], voice morphing [2, 3, 4], voice transposition [5], or voice enhancement for speakers with vocal disorders [6].

Human speech production is often modeled as a source-filter system [7]. For voiced sounds the source signal may be modeled as a pseudo-periodic pulse train, resulting from the vibration of vocal folds. In such cases, the excitation determines the pitch f_0 (perceived fundamental frequency) as well as other characteristics such as breathness or falsetto emission [8, 9]. Unvoiced sounds are generated without vocal folds vibration. In these cases, the source models the turbulent behavior of the air flow as a noise signal. The filter is responsible for the distinction between phonemes and for the speaker's timbre, though the glottal excitation may influence the timbre too. In frequency domain, voiced speech segments are represented by a train of impulses spaced by f_o with their amplitudes multiplied by the filter's spectral envelope.

Pitch shifting was initially implemented by speed changes in musical recordings. Although this kind of approach was successful for some instruments, it was not possible to change the pitch without changing the duration of the signal. For speech signals, this technique worked very poorly, since it shifted the entire original spectrum, resulting in a very unnatural human voice. Successful experiences with pitch modification in speech result from parametric coding techniques [10, 11], that use the source-filter model discussed above, and from FFT based methods, like the *Phase Vocoder*, that is subject to *phasiness* distortion [12]. A family of non-parametric techniques include the *pitch synchronous overlap-and-add* (PSOLA) and its variations [13]. The PSOLA method segments the signal at pitch periods, then overlaps-and-adds them back to synthesize the output signal with the desired pitch characteristics. Extensions of this technique include combinations with the linear prediction and FFT approaches (LP- and FD-PSOLA) [13, 14].

This paper deals with the pitch shifting problem by using a sequential approach to LP-PSOLA. Instead of classical block techniques [11], it uses the recursive least-squares (RLS) adaptive filter to estimate the LP model in a sample-by-sample manner. This leads to a more natural sounding synthesized signal with smoother variations of the LP model than classical block techniques. Parameterization of PSOLA techniques adds flexibility towards further improvements and different applications.

The paper is organized as follows: the overall pitch-modifying system is presented in Section 2, which comprises a description of the RLS-based LP modeling and the excitation synthesis using LP-PSOLA; experimental results illustrating the system performance are included in Section 3; conclusions and future developments are pointed out in Section 4.

2. PROPOSED SYSTEM

2.1. LP modeling with RLS algorithm

In the LP model, depicted in Figure 1, the **a** FIR-filter coefficients are used to estimate the voice signal spectral envelope, and the prediction residual $e[n]$ is used to recover the original source signal.

In the proposed scheme, the RLS adaptive algorithm is employed to determine the **a** coefficients that minimize the objective function

$$\xi_{RLS}[n] = \sum_{i=0}^{n} \lambda^{n-i} e^2[i] = \sum_{i=0}^{n} \lambda^{n-i} \left(s[i] - \widehat{s}[i]\right)^2, \quad (1)$$

where $0 \ll \lambda < 1$ is the so-called forgetting factor and

$$\widehat{s}[n] = \sum_{p=1}^{P} a_p s[n-p] = \mathbf{a}^T \mathbf{s}[n-1], \quad (2)$$

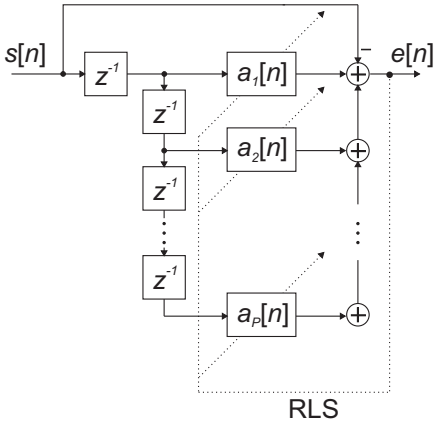


Figure 1: Block diagram of the LP scheme.

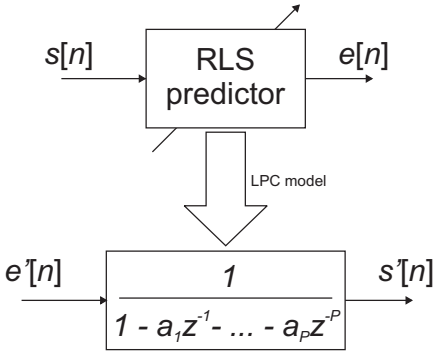


Figure 2: Analysis and synthesis modes using the RLS-LP model.

where

$$\mathbf{s}[n-1] = \left[\begin{array}{cccc} s[n-1] & s[n-2] & \ldots & s[n-P] \end{array}\right]^T, \quad (3)$$

$$\mathbf{a} = \left[\begin{array}{cccc} a_1 & a_2 & \ldots & a_P \end{array}\right]^T. \quad (4)$$

The result is a coefficient vector given by

$$\mathbf{a}[n] = \mathbf{R}^{-1}[n-1]\,\mathbf{p}[n] \quad (5)$$

where

$$\mathbf{R}[n-1] = \sum_{i=0}^{n} \lambda^{n-i}\mathbf{s}[i-1]\mathbf{s}[i-1], \quad (6)$$

$$\mathbf{p}[n] = \sum_{i=0}^{n} \lambda^{n-i}\mathbf{s}[i-1]s[i]. \quad (7)$$

The values of $\mathbf{R}^{-1}[n-1]$ and $\mathbf{p}[n]$ in Equation (5) can be calculated in a recursive way, to avoid extra computational burden, as given by

$$\mathbf{p}[n] = \mathbf{s}[n-1]s[n] + \lambda\mathbf{p}[n-1], \quad (8)$$

$$\mathbf{R}^{-1}[n-1] = \frac{1}{\lambda}\left[\mathbf{R}^{-1}[n-2] - \frac{\Psi[n]\Psi^T[n]}{\lambda + \Psi^T[n]\mathbf{s}[n-1]}\right], \quad (9)$$

where

$$\Psi[n] = \mathbf{R}^{-1}[n-2]\mathbf{s}[n-1]. \quad (10)$$

For further details on the RLS implementation the reader may refer to [15].

Using the RLS coefficients obtained in the analysis cycle, the LP model can be employed in the synthesis cycle with a new excitation signal $e'[n]$ to get the modified signal $s'[n]$ with the desired pitch characteristics, such that

$$s'[n] = e'[n] - \mathbf{a}^T[n]\mathbf{s}'[n-1], \quad (11)$$

as represented in Figure 2. The signal $e'[n]$ is obtained from $e[n]$ using PSOLA algorithm as detailed in Section 2.2.

One may notice that the RLS-LP model is determined for each time sample n, leading to smooth transitions between consecutive models. The resulting model quality depends on the choices of the number of LP coefficients P and the forgetting factor λ. It is possible to show that the RLS sequential solution is a special case of the classical LP block solutions, with λ controlling the equivalent length of an exponential analysis window [16]. Proper values of P and λ may vary for distinct sampling rates.

2.2. Source implementation

There are several approaches to generate the excitation signal $e'[n]$. The most straightforward is to use an impulse train plus noise for voiced segments, but it often leads to artificial speech results. Several works employ a glottal pulse model [8, 9, 17, 18] to emulate the vocal effort, the parameterization of which constitutes a cumbersome task. One sample of glottal pulse from $e[n]$ can also be used as a pulse model to generate $e'[n]$ [16].

In this work, the LP error $e[n]$ is used to generate the modified excitation signal $e'[n]$, as illustrated in Figure 2. The desired pitch is introduced in $e'[n]$ by means of a PSOLA technique applied to the residual signal $e[n]$. This procedure constitutes the so-called LP-PSOLA technique [13]. To do that, one applies a peak tracking algorithm to determine the instants of glottal closure, which correspond to changes in the statistical properties of the signal. Figure 3 illustrates the result of a pitch marking procedure on a speech signal $s[n]$. In this figure, $p_m[n]$ indicates the pitch marks and the intervals $p[n]$ are the corresponding pitch periods. The procedure can be implemented using wavelets [19], by observing variations

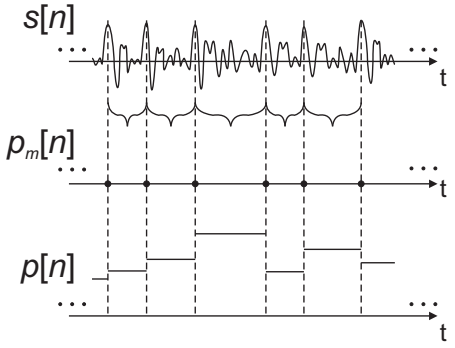


Figure 3: Pitch marks $p_m[n]$ and associated pitch detection $p[n]$ of a speech signal $s[n]$.

of statistical properties [20], or simply by direct observation of the amplitude envelope.

Once reliable pitch marks $p_m[n]$ and pitch periods $p[n]$ of the original signal are determined, the desired pitch contour can be modified as desired. For that purpose, new pitch marks $p'_m[n]$ are determined corresponding to a new pitch period $p'[n]$, such that

$$p'[n] = \beta[n]p[n], \tag{12}$$

where $\beta[n]$ is the pitch-period modification factor, which can be made variable for natural prosody modification, automatic pitch correction, vibrato synthesis, and so on. The new pitch marks $p'_m[n]$ are determined by forcing an interval of $p'[n]$ samples between two consecutive marks, such that a pitch mark will be placed at position $n+p'[n]$ if n has a pitch mark (*i.e.* $p'_m\,[n+p'[n]] = 1$ if $p'_m[n] = 1$, where pitch mark positions are indicated by 1).

The next step is to link each new pitch mark $p'_m[n]$ with its corresponding closest peak on the original signal $p_m[n]$. This is done straightforwardly by comparing the time index of $p_m[n]$ and $p'_m[n]$, as illustrated in Figure 4.

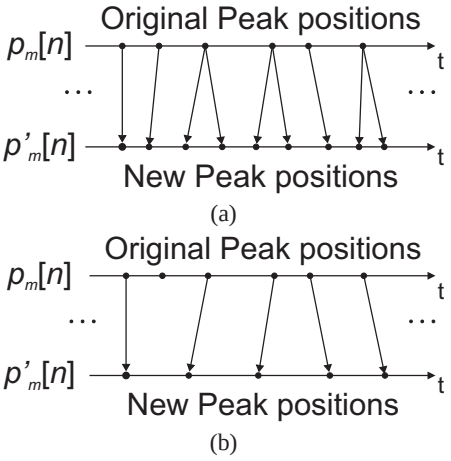


Figure 4: Pitch mark association for the synthesized source signal with: (a) Increased pitch; (b) Decreased pitch.

In the final step of the new source generation, each peak in the original signal is then segmented, by two half-hanning windows, starting at the preceding pitch mark and ending at the next one. The resulting source segments are put together by an overlap-and-add procedure according to the new pitch period $p'[n]$ obtained previously, as given in Figure 5.

3. EXPERIMENTAL RESULTS

This section describes some practical experiments performed with the proposed pitch-modification system.

Example 1: A portion of a song recorded by a female Brazilian singer was modified using $\beta[n] = 2$ and $\beta[n] = 0.5$. Figure 6 shows a small portion of the original and modified signals, whereas Figures 7 shows their corresponding spectrograms.

The proper pitch modification can be inferred from Figure 6 by noticing how the modified peaks become more separated or closer when $\beta[n] = 2$ and $\beta[n] = 0.5$, respectively. A similar conclusion

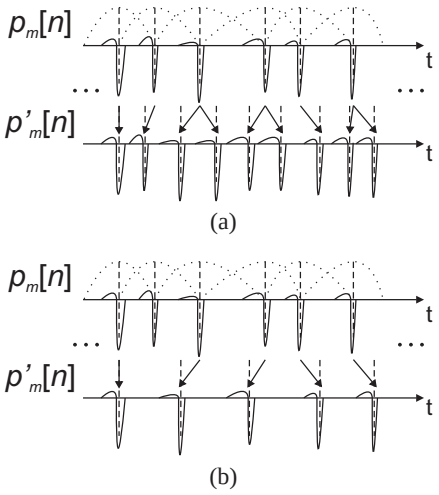


Figure 5: Composition of the new source signal with: (a) Increased pitch; (b) Decreased pitch. In each case, the dashed lines correspond to the segmentation windows centered at each original pitch mark.

can be drawn observing the fringes in the modified spectrograms in Figure 7.

Example 2: Once again the pitch characteristic of a song was modified with $\beta[n] = 2$ and $\beta[n] = 0.5$. This time, however, the recorded voice of a male singer was employed. The time- and frequency-domain representations of the resulting signals are depicted in Figures 8 and 9, respectively.

Once again, from these figures, it is easy to identify the desired pitch modification, while the original spectral envelope is kept essentially unchanged in all cases.

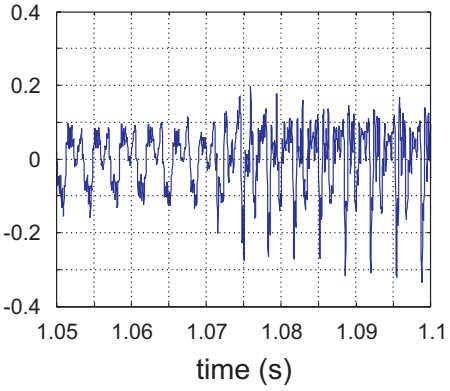
Example 3: Figure 10 compares the results of the proposed method and PSOLA for a one-octave decreasing of pitch, *i.e.* $\beta[n] = 2$, on the same signal employed in Example 1. This figure illustrates a major drawback of the PSOLA algorithm, which is the significant energy decrease in between consecutive peak marks when $\beta[n] > 1$. Although larger analysis windows could be employed in such cases, they could lead to spurious peaks on the synthesized signal, since adjacent peaks would not be sufficiently reduced by the analysis windows. These spurious peaks might lead to roughness on the modified signal.

It is worth noting that, instead of directly overlapping portions of the output signal as in PSOLA, the LP-PSOLA intrinsically keeps the responses to each excitation pulse individualized, as in the voice production model itself. This feature, coupled with the smoothly tracked RLS-LP model, allows one to expect that the proposed system can produce a more natural synthesized voice.

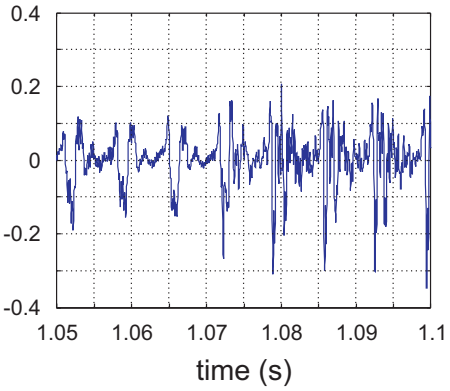
4. CONCLUSIONS

A complete system for pitch modification of voice signals was presented in this paper. Spectral envelope modeling is performed by an adaptive RLS filter, leading to a sample-by-sample estimation of the LP model. This results in smooth transitions in the estimated model, thus yielding a more natural synthesized signal.

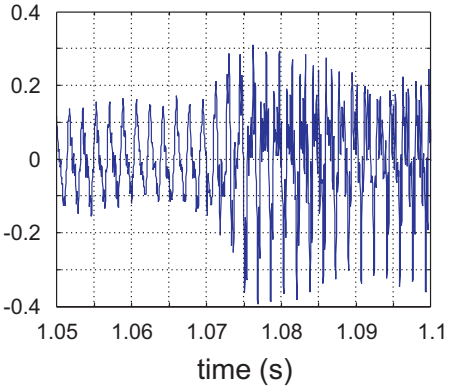
The source signal with the desired pitch characteristics is ob-



(a)

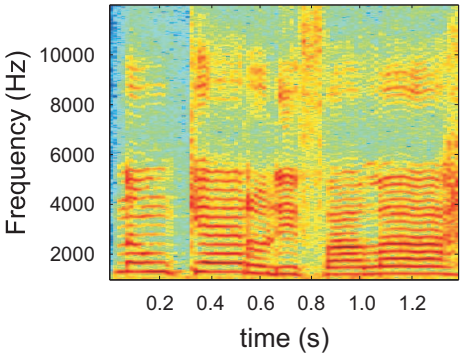


(b)

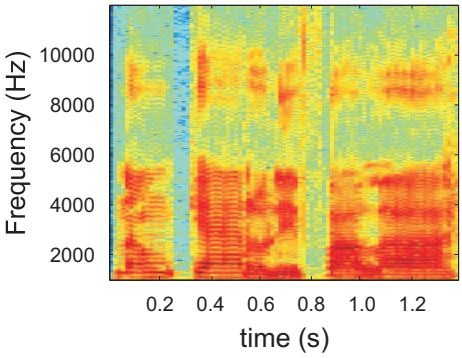


(c)

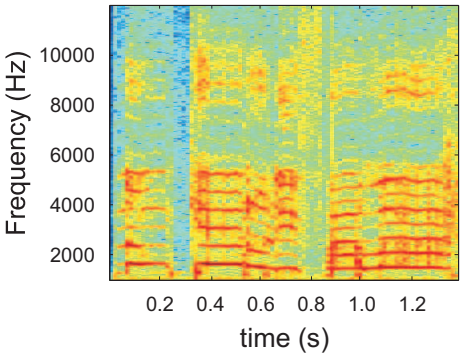
Figure 6: Extracts of signals in Example 1: (a) Original signal; (b) Modified signal with $\beta[n] = 2$; (c) Modified signal with $\beta[n] = 0.5$.



(a)



(b)



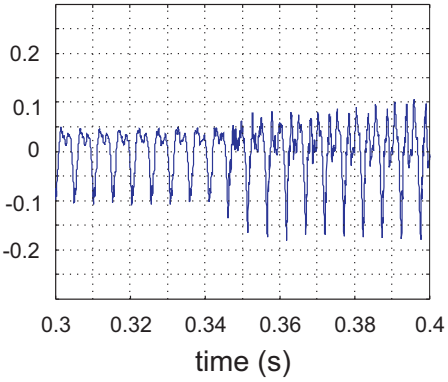
(c)

Figure 7: Spectrograms of signals in Example 1 showing that the original spectral envelope is preserved in all cases: (a) Original spectrogram; (b) Modified spectrogram with $\beta[n] = 2$; (c) Modified spectrogram with $\beta[n] = 0.5$.

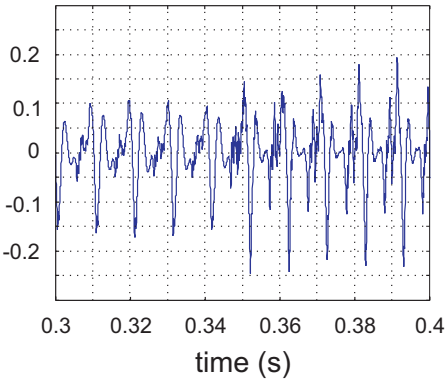
tained by applying a PSOLA algorithm on the RLS residual error. The advantage of this method is to preserve the individuality of each glottal pulse. Furthermore, in case of insufficient LP-model order, part of the spectral envelope information is carried by the RLS residual error, thus reducing the envelope modeling error in the synthesized signal. Additionally, the proposed system is able to sustain signal information in between pitch marks even for large pitch-modification factors $\beta[n]$.

Informal subjective tests have shown good results for pitch scale modification in the range $0.5 \leq \beta[n] \leq 2$, which, in musical terms, means from an octave downwards to an octave upwards. This system is currently being tested against standard pitch modification methods.

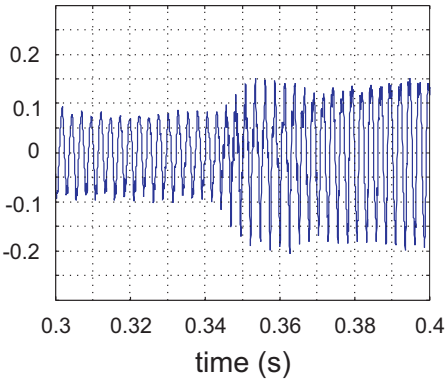
The parameterization inherent to the described method suggests the system can be extended to fit voice *morphing* applications, *e.g.* male-female conversion, voice transposition, and non-human voice synthesis for voice editing in cartoons.



(a)

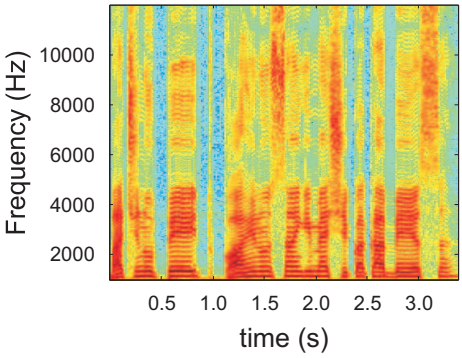


(b)

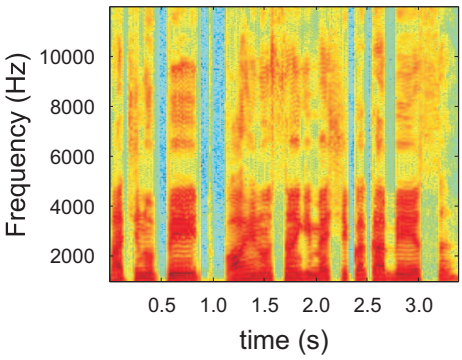


(c)

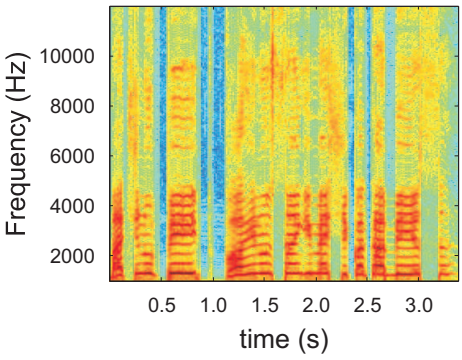
Figure 8: Extracts of signals in Example 2: (a) Original signal; (b) Modified signal with $\beta[n] = 2$; (c) Modified signal with $\beta[n] = 0.5$.



(a)



(b)



(c)

Figure 9: Spectrograms of signals in Example 1 showing that the original spectral envelope is preserved in all cases: (a) Original spectrogram; (b) Modified spectrogram with $\beta[n] = 2$; (c) Modified spectrogram with $\beta[n] = 0.5$.

5. ACKNOWLEDGMENTS

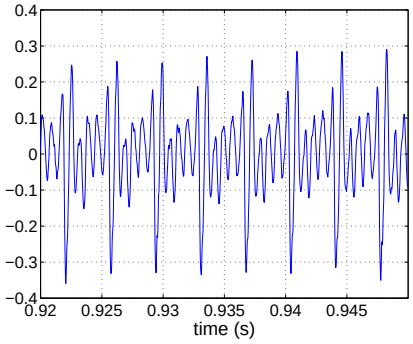
The authors would like to thank the Brazilian sponsors of this work, *Conselho Nacional de Desenvolvimento Científico e Tecnológico* (CNPq) and *Fundação Carlos Chagas Filho de Amparo à Pesquisa do Estado do Rio de Janeiro* (FAPERJ), for their support.

6. REFERENCES

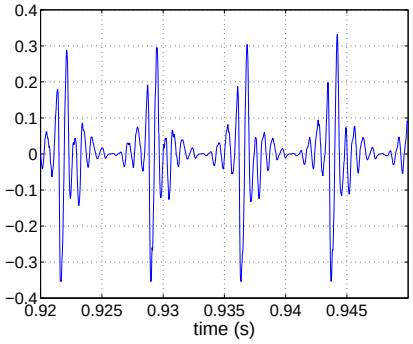
[1] J. Bonada and X. Serra, "Synthesis of the singing voice by performance sampling and spectral models," *IEEE Signal Processing Magazine*, vol. 24, no. 2, pp. 67–79, March 2007.

[2] L. Fabig and J. Janer, "Transforming singing voice expression - the sweetness effect," in *Proc. of the DAFx04 - 7th International Conference on Digital Audio Effects*, Naples, Italy, October 2004.

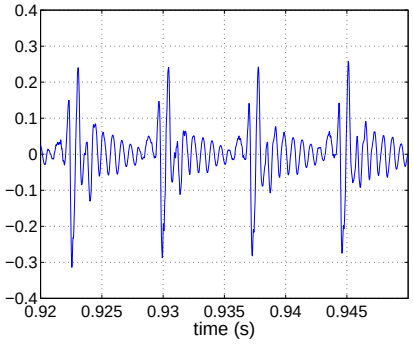
[3] A. Loscos and J. Bonada, "Emulating rough and growl voice



(a)



(b)



(c)

Figure 10: Extracts of signals in Example 3: (a) Original signal; (b) Signal modified by PSOLA; (c) Signal modified by the proposed method.

in spectral domain," in *Proc. of the DAFx04 - 7th International Conference on Digital Audio Effects*, Naples, Italy, October 2006.

[4] P. Depalle, G. Garcia, and X. Rodet, "The recreation of a castrato voice, farinelli's voice," in *Proc. of the WASPAA'95 - Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, USA, October 1995, IEEE, pp. 15–18.

[5] E. Turajlic, D. Rentzos, S. Vaseghi, and C. H. Ho, "Evaluation of methods for parameteric formant transformation in voice conversion," in *Proc. of the ICASSP'03 - International Conference on Acoustics, Speech, and Signal Processing*, Hong Kong, Hong Kong, April 2003, IEEE, pp. 724–727.

[6] J. Bonada and A. Loscos, "Esophageal voice enhancement by modeling radiated pulses in frequency domain," *121st Audio Engineering Society Convention*, October 2006, Preprint 6952.

[7] J. R. Deller, J. H. L. Hansen, and J. G. Proakis, *Discrete-Time Processing of Speech Signals*, Wiley-IEEE, 1999.

[8] Q. Fu and P. Murphy, "Robust glottal source estimation based on joint source-filter model optimization," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 14, no. 2, pp. 492–501, March 2006.

[9] D. H. Klatt and L. C. Klatt, "Analysis, synthesis, and perception of voice quality variations among female and male talkers," *Journal of the Acoustical Society of America*, vol. 87, no. 2, pp. 820–857, February 1990.

[10] J. A. Moorer, "The use of linear prediction of speech in computer music applications," *Journal of Audio Engineering Society JAES*, vol. 27, no. 3, pp. 134–140, March 1979.

[11] J. Makhoul, "Linear prediction: a tutorial review," *Proceedings of the IEEE*, vol. 63, no. 4, pp. 561–580, April 1975.

[12] J. Laroche and M. Dolson, "Phase-vocoder: about this phasiness business," in *Proc. of the WASPAA'97 - Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, USA, October 1997, IEEE.

[13] E. Moulines and J. Laroche, "Non-parametric techniques for pitch-scale and time-scale modification of speech," *Speech Communication*, vol. 16, no. 2, pp. 175–205, February 1995.

[14] K. S. Rao and B. Yegnanarayana, "Prosody modification using instants of significant excitation," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 14, no. 3, pp. 972–980, May 2006.

[15] P. S. R. Diniz, *Adaptive Filtering: Algorithms and Practical Implementations*, Kluwer, 2 edition, 2002.

[16] R. C. D. de Paiva, L. W. P. Biscainho, and S. L. Netto, "A sequential system for voice pitch modification," in *Proc. of the AES-Brazil'07, 5th AES-Brazil Conference*, São Paulo, Brazil, May 2007, Audio Engineering Society, pp. 11 – 16.

[17] G. Fant, J. Liljencrants, and Q. Lin, "A four-parameter model of glottal flow," STL-QPSR 26 4, Dept. for Speech, Music and Hearing - Royal Institute of Technology, Stockholm, Sweden, 1985.

[18] H. L. Lu, *Toward a High-Quality Singing Synthesizer with Vocal Texture Control*, Ph.D. thesis, Dept. of Electrical Engineering - Stanford University, Palo Alto, USA, July 2002.

[19] S. Kadambe and G. F. Boudreaux-Bartels, "Application of the wavelet transform for pitch detection of speech signals," *IEEE Transactions on Information Theory*, vol. 38, no. 2, pp. 917–924, March 1992.

[20] C. Ma, Y. Kamp, and L. F. Willems, "A Frobenius norm approach to glottal closure detection from the speech signal," *IEEE Transactions on Speech and Audio Processing*, vol. 2, no. 2, pp. 258–265, April 1994.

SINUSOID MODELING IN A HARMONIC CONTEXT

Wen Xue, Mark Sandler

Centre for Digital Music, Dept. Elec. Eng.,
Queen Mary, University of London, UK
{xue.wen,mark.sandler}@elec.qmul.ac.uk

ABSTRACT

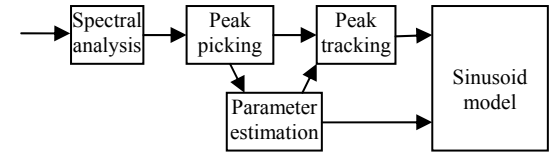
This article discusses harmonic sinusoid modeling. Unlike standard sinusoid analyzers, the harmonic sinusoid analyzer keeps close watch on partial harmony from an early stage of modeling, therefore guarantees the harmonic relationship among the sinusoids. The key element in harmonic sinusoid modeling is the harmonic sinusoid particle, which can be found by grouping short-time sinusoids. Instead of tracking short-time sinusoids, the harmonic tracker operates on harmonic particles directly. To express harmonic partial frequencies in a compact and robust form, we have developed an inequality-based representation with adjustable tolerance on frequency errors and inharmonicity, which is used in both the grouping and tracking stages. Frequency and amplitude continuity criteria are considered for tracking purpose. Numerical simulations are performed on simple synthesized signals.

1. INTRODUCTION

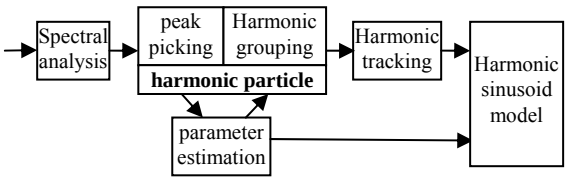
The standard sinusoid model [1,2] expresses an audio signal as the combination of slow-varying sinusoids plus a noise. Although the sinusoids clearly model the partials of pitched sounds, it has not been made explicit. Due to the lack of emphasis on the relationship among partials, the standard sinusoid tracking methods cannot guarantee harmonic consistence. Accordingly, the results do not provide a solid base for extracting pitched events. On the other hand, matching pursuit based methods have been proposed to extract harmonic structure from music [3, 4]. However, these methods lack the freedom of representing time-varying frequency within a single object, and tend to represent a harmonic event with time-varying pitch as multiple events. To overcome these difficulties, we apply the harmonic constraint, which is more flexible than matching pursuits, in an early stage of sinusoid analysis, preferably before the tracking of partials. This upgrades sinusoid modeling to *harmonic sinusoid modeling*. The frameworks of sinusoid and harmonic sinusoid analyzers are compared in Figure 1. The key element of the sinusoid model, the *short-time sinusoid atom*, becomes *harmonic particle*. The two main parts of the sinusoid analyzer, i.e. the sinusoid detector and the partial tracker, are replaced by harmonic particle detector and harmonic sinusoid tracker, respectively. Compared to standard sinusoid models, the harmonic model provides a higher-level description of pitched events, which enables an extensive range of analysis and synthesis operations.

A harmonic sinusoid is described by sinusoidal parameters $\{ f_l^m, a_l^m, \varphi_l^m \mid 0 \le l < L,\ 1 \le m \le M \}$, where L is the number of frames, M is the number of partials, f_l^m (a_l^m, φ_l^m) is the instantaneous frequency (amplitude, phase angle) of the m^{th} partial at the l^{th} frame. By fixing m we get a description of the m^{th} partial; by fixing l we get a description of the harmonic particle at the l^{th} frame.

This article is arranged as follows. Section 2 discusses the grouping of sinusoid atoms into harmonic particles. Section 3 discusses the harmonic sinusoid tracker. Section 4 presents some numerical results of the algorithms, followed by a brief conclusion in section 5.



(a) Sinusoid analysis



(b) Harmonic sinusoid analysis

Figure 1:*Comparing standard and harmonic sinusoid analyzers*

2. HARMONIC GROUPING WITH INEQUALITIES

We assume that the short-time sinusoid atoms have already been detected at frame *l*. This is accomplished by spectral peak picking [1] and sinusoid parameter estimation [2,5]. The harmonic grouping module collects sinusoid atoms, whose frequencies can be regarded as harmonic, from this pre-detected set. Perfect harmony is characterized by all partial frequencies being multiples of a fundamental frequency. Let f^m be the frequency of the m^{th} partial, then perfect harmony implies $f^m = mf^1$. However, this does not provide a practical way to spot harmonic particles from the pre-detected peaks, mainly for two reasons: 1) the frequency estimates carry errors, and 2) perfect harmony is not always guaranteed.

2.1. Inharmonicity

The 2nd problem, known as inharmonicity, is best known for free-vibrating strings. [6] gives an example of explicitly expressing the partial frequencies as a function of fundamental frequency f^1 and a stiffness coefficient B:

$$f^m(f^1, B) = mf^1 \cdot [1 + B(m^2 - 1)]^{1/2} \qquad (1)$$

B is a constant for a given string. Strictly speaking (1) only approximately describes the inharmonicity due to string stiffness [7], and may still carry an error. However, it is reasonable to assume that this error is so small that it can be "absorbed" into the frequency estimation error.

2.2. Frequency estimation error

The frequency estimate of a partial can be very accurate when its pitch is stable and the partial is spared of noise and disturbance. In real-world recordings, however, the pitch may have smooth or repetitive variations fast enough to affect frequency estimates, and noise and concurrent sinusoids do disturb sinusoid analyzers. The frequency estimate error depends on the estimator type and the signal behaviour, the latter being highly unpredictable. The estimation of the error bound is out of the scope of this paper. However, we always assume that we can find an error bound Δ^m for f^m. Let the frequency estimate be $\hat{f}^m$, then

$$| \hat{f}^m - f^m | < \Delta^m \tag{2}$$

The error bound Δ^m does not have to be tight. In most cases it is reasonable to set Δ^m at 1 spectral bin for low partials, and a few more for high partials if the pitch variation is fast.

Combining (1) and (2) we get

$$\hat{f}^m - \Delta^m < m f^1 \cdot [1 + B(m^2 - 1)]^{1/2} < \hat{f}^m + \Delta^m \tag{3a}$$

Equation (3) relates the frequency estimates to the two parameters of the partial frequency model (1), i.e. f^1 and B. If the frequency estimate satisfies (3a) for some f^1 and B, we allow it to be the m^{th} partial for this f^1-B pair.

2.3. Harmonic partial frequencies

Now we address the following problem: given frequency estimates of M partials, $\hat{f}^{m_1}$, $\hat{f}^{m_2}$, …, $\hat{f}^{m_M}$, where m_1^{th}, m_2^{th}, …, m_M^{th} are the partial indices, can they be grouped as harmonic partials? The answer is straightforward: if there exists f^1 and B so that (3a) holds for all the frequency estimates, then they can be regarded as harmonic partials. In other words, let the solution set of

$$\begin{cases} \hat{f}^{m_1} - \Delta^{m_1} < m_1 f^1 \sqrt{1 + B(m_1^2 - 1)} < \hat{f}^{m_1} + \Delta^{m_1} \\ \hat{f}^{m_2} - \Delta^{m_2} < m_2 f^1 \sqrt{1 + B(m_2^2 - 1)} < \hat{f}^{m_2} + \Delta^{m_2} \\ \quad \cdots\cdots \\ \hat{f}^{m_M} - \Delta^{m_M} < m_M f^1 \sqrt{1 + B(m_M^2 - 1)} < \hat{f}^{m_M} + \Delta^{m_M} \end{cases} \tag{3b}$$

be R, then the given frequencies can be regarded as harmonic partials if and only if R≠Φ. However, (3b) is a non-linear inequality system, which makes R hard to represent in the f^1-B plane. We linearize (3b) using the substitutions

$$F = (f^1)^2, \; G = FB = B(f^1)^2 \tag{4}$$

then (3b) becomes

$$\begin{cases} g_{m_1-} < F + k_1 G < g_{m_1+} \\ g_{m_2-} < F + k_2 G < g_{m_2+} \\ \quad \cdots\cdots \\ g_{m_M-} < F + k_M G < g_{m_M+} \end{cases} \tag{5}$$

where $g_{m-} = \left(\dfrac{\hat{f}^m - \Delta^m}{m} \right)^2$, $g_{m+} = \left(\dfrac{\hat{f}^m + \Delta^m}{m} \right)^2$, $k = m^2 - 1$. The solution of (5) is R in the F-G plane. We impose additional constraints on the allowable ranges for f^1 (e.g. 0~0.5) and B (e,g 0~0.001). These constraints are linear in the F-G plane. R, if not empty, is always a convex polygon. We represent R using a list of its N vertices in the (F-G) plane, i.e. {N; (F_n, G_n), n=0, 1, …, N-1}. To solve for R we initialize it by presetting the f^1 and B ranges (so R is a close polygon from the beginning), and apply the constraints one after another. Each constraint chops off the part of R outside a pair of parallel lines specified by $F + k_m G = g_{m\pm}$. The more partial frequencies are used, the smaller becomes R.

R represents a range for f^1 and B so that those points, and only those points in R, can be the f^1-B pairs to associate the given frequencies with. We directly have

$$\sqrt{\min_n F_n} < f^1 < \sqrt{\max_n F_n}, \; \min_n \frac{G_n}{F_n} < B < \max_n \frac{G_n}{F_n} \tag{6}$$

That is, the span of R on the F axis determines the precision of f^1, and the angular sweep of R, with respect to (0,0), determines the precision of B. The smaller R is, the more precise are f^1 and B. The m^{th} partial frequency is located by

$$f_-^m(R) < f^m < f_+^m(R), \tag{7a}$$

where

$$f_-^m(R) = m\sqrt{\min_n (F_n + kG_n)}, \; f_+^m(R) = m\sqrt{\max_n (F_n + kG_n)}. \tag{7b}$$

(7a) provides an estimation of f^m with a better precision than Δ^m. However, most of the time we need (7a) for judging whether $\hat{f}^m$ is compatible with R. If R is derived without using the m^{th} partial, then $\hat{f}^m$ can be regarded as an additional harmonic partial, as long as

$$f_-^m(R) - \Delta^{m_1} < \hat{f}^m < f_+^m(R) + \Delta^{m_1}. \tag{7c}$$

2.4. Grouping partials by harmony

The partial grouping based on the inequality representation R is simple in principle. To find a harmonic particle, one

1) initialize R, find the first partial;
2) use the found partial to update R;
3) use R to compute the range to look for the next partial;
4) find the next partial;
5) if there are still partials to find, go back to 2.

Notice that here the "first partial" refers to the first *available* partial: it does not have to be the fundamental partial, but may be any partial whose partial index is known. Figure 2 shows how R is updated for a perfect harmonic particle with neither frequency estimation error nor spurious peaks. We choose f^1=0.1 and Δ^m=0.01, 0≤B≤0.05. R obtained by using the lowest 1, 2, 3, 4 partials are shown in (a), (b), (c), (d) respectively.

In more practical cases there are three complications. First, we do not have a range to look for the first partial; second, correct partials may not appear as a spectral peak, and therefore cannot be located; third, multiple partials may be found in step 4. We discuss them in reverse order.

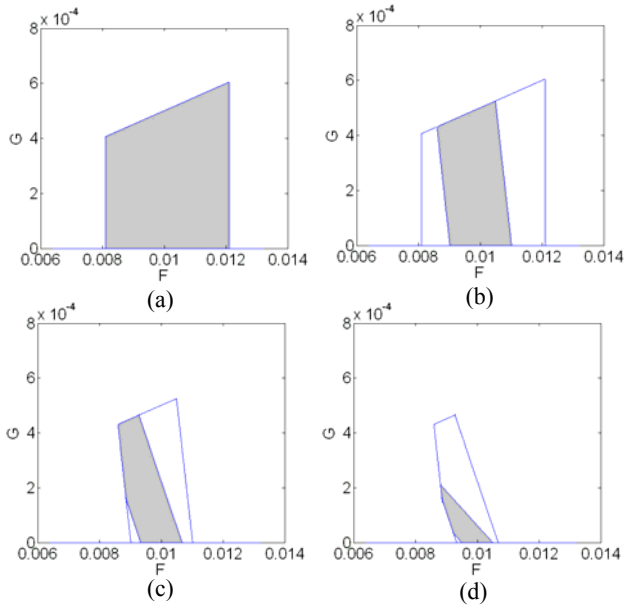


Figure 2: *Updating R using found partials*

2.4.1. Competing peaks

If more than one atom lies in the searching range for the m^{th} partial, they become competing peaks. A peak that competes with the true partial may be either a spurious peak, or a partial of a concurrent event. We can derive a *candidate* harmonic particle from every peak, and choose one from these candidate particles that is optimal in some sense. To do this we need a criterion, i.e. a scoring function, to compare two harmonic particles. The strength-harmony criterion is based on two assumptions:

(1) most spurious peaks are weak;

(2) correctly captured partials tend to have less departure from the model frequencies.

From assumption (1) we derive the strength criterion. If the strength of particle p_1 is higher than that of particle p_2, then p_1 is given a higher score on the strength side. The score can be the total amplitude calculated by summing up partial amplitudes, or the total partial SPL calculated by summing up the logarithms of partial amplitudes, or some other more perceptual measures. We always assume it can be written in an additive form, i.e.

$$s_a\left(\{\hat{a}^m\}_{m=1,2,\cdots}\right)=\sum_m s_a(\hat{a}^m)\text{ , }\ s_a(\hat{a}^m)\ge 0\text{ , }\ s'_a(\hat{a}^m)>0\ . \qquad (8a)$$

Assumption (2) favours partials with more predictable frequencies. As said in 2.2, the error bounds Δ^m used for harmonic grouping are not tight. Using large error bounds provides good robustness against frequency estimation errors. However, it is a main reason that we have competing peaks. To make up for this, we introduce the harmony criterion based on the departure of frequency estimates from the model. The departure of the m^{th} partial frequency estimate $\hat{f}^m$ from model R is

$$d(\hat{f}^m, m, R)=\begin{cases} f_-^m(R)-\hat{f}^m, & \hat{f}^m < f_-^m(R) \\ 0, & f_-^m(R)\le \hat{f}^m \le f_+^m(R) \\ \hat{f}^m - f_+^m(R), & \hat{f}^m > f_+^m(R) \end{cases}\ . \qquad (9)$$

where $f_-^m(R)$ and $f_+^m(R)$ are defined by (7b). We also assume that the harmony score s_f can be written in an additive form:

$$s_f(\{\hat{f}^m\}_{m=1,2,\cdots},R)=\sum_m s_f^m\left(d(\hat{f}^m,m,R),s_a(\hat{a}^m)\right) \qquad (10)$$

The dependency of s_f^m on $s_a(\hat{a}^m)$ allows us to design a harmony score that is consistent with $s_a(\hat{a}^m)$ in some sense. Let Δ^m be a maximal allowable frequency departure. We choose to assign a 100% penalty to $s_a(\hat{a}^m)$ if $d(\hat{f}^m,R)\ge\Delta^m$, and no penalty if $d(\hat{f}^m,R)=0$, i.e.

$$s_f^m(d,s)=\begin{cases} 0, & d=0 \\ -sd/\Delta^m, & 0<d<\Delta^m \\ -s, & d\ge\Delta^m \end{cases} \qquad (11)$$

Between $d(\hat{f}^m,R)=0$ and $d(\hat{f}^m,R)\ge\Delta^m$ we assign a partial penalty, like the linear function in (11). The final score for evaluating a harmonic particle is

$$s\left(\{\hat{a}^m,\hat{f}^m,R\}_{m=1,2,\cdots}\right)=s_a\left(\{\hat{a}^m\}_{m=1,2,\cdots}\right)+s_f(\{\hat{f}^m\}_{m=1,2,\cdots},R) \qquad (12)$$

The number of candidates grows whenever we have competing peaks. However, at any stage we can combine two candidates p_1 and p_2, if 1) $s_1>s_2$ and 2) $R_1\supseteq R_2$. In particular, if the two peaks with frequency estimates $\hat{f}_1^m$ and $\hat{f}_2^m$ are competing for the m^{th} partial, $s_a(\hat{a}_1^m)+s_f^m(\hat{f}_1^m,R)>s_a(\hat{a}_2^m)+s_f^m(\hat{f}_2^m,R)$, then we can immediately discard candidate 2 if a) $\hat{f}_1^m>\hat{f}_2^m$ and $\hat{f}_1^m-\Delta^m\le f_-^m(R)$, or b) $\hat{f}_1^m<\hat{f}_2^m$ and $\hat{f}_1^m+\Delta^m\ge f_+^m(R)$. Finally the candidate harmonic particle with the highest score is selected.

2.4.2. Unfound partials

In addition to the spurious partial problem, a true partial may fail to appear as a spectral peak if 1) it is too weak, or 2) it is masked by noise. A partial being unfound is not a problem by itself, as its absence does not affect R or the searching of other partials. The real problem is that we do not know whether a partial appears as a spectral peak or not. Even when a partial does not produce a peak, it is possible for spurious peaks to appear where the partial is expected. If this were the case and the spurious peak were used to update R, the searching range of further partials would be biased. A safe way to deal with the unfound partial problem is to always reserve a candidate for "unfound partial", even when one or more atom have been located. In fact this is necessary only when the size of R is relatively large and the located atom has large frequency departure, in which case it substantially reduces the size of R. Unfound partials do not contribute to $s_a(\hat{a}_1^m)$ or $s_f^m(\hat{f}_1^m,R)$.

2.4.3. Unknown range for the first partial

The frequency range to look for a partial is calculated from R. Once the first partial has been located, R can be updated with its frequency estimates so that the search range for any further partial is reduced to no more than a few bins. This, however, does not apply to the first partial. In many cases a small frequency range of the first partial can be provided externality (s.a. by a pitch detector, a score, or a user), or by a harmonic particle in an adjacent frame during the tracking stage (see section 3). However, if there is no

such information available, we can run an exhaustive search through the pre-detected atoms. That is, we start with a strong peak, assume this is the 1st, 2nd, …, m^{th}, … partial, and derive a harmonic particle candidate from each assumption; then we compare these candidates with some criterion to choose the best one. If the audio frame has a single pitch, the found particle shall represent this pitched event. If the audio frame has multiple pitches, the found particle is interpreted as a *predominant harmonic particle*, representing one of the pitched events.

2.5. Estimating f^1 and B

The polygon R represents our knowledge of f^1 and B accumulated from the frequency estimates involved in (3b). f^1 and B do not appear explicitly during harmonic grouping or harmonic partial tracking. (6) estimates the two parameters as intervals. The sizes of the intervals are determined by the frequency estimates $\hat{f}_1^m$ and the error bounds Δ^m. As mentioned before, we use relatively large error bounds to enhance robustness. This results in a large R and imprecise f^1 and B. Accordingly, more precise estimates of f^1 and B can be obtained by reducing the overlarge error bounds. Let θ be a number between 0 and 1. By setting the error bound associated with the m^{th} partial to $\theta\Delta^m$, we can get an f^1-B range R(θ). Apparently R(1)=R, and the size of R(θ) (hence the precision of f^1 and B) is monotonous regarding θ. Since the size of R(0) is 0, we know that there exists η, 0≤η<1, so that the size of R(η) is 0, and ∀θ>η, the size of R(θ) is positive. In other words, by reducing θ from 1 to η, we shrink R(θ) from R to a zero-sized polygon. We can further prove this zero-sized polygon to be a single point. Therefore R(η) provides estimates of f^1 and B in the precise form.

We consider the constraints (3b) with argument θ. Given a point (f^1, B)∈R, it lies on R(θ) if and only if it satisfies the constraint

$$f^m(f^1,B)-\theta\Delta^m < \hat{f}^m < f^m(f^1,B)+\theta\Delta^m,\ \forall m \qquad (13a)$$

or

$$\theta > \max_m \frac{\left|\hat{f}^m - f^m(f^1,B)\right|}{\Delta^m} \equiv \theta(f^1,B) \qquad (13b)$$

θ(f^1, B) is the minimal value of θ for R(θ) to contain the point (f^1, B). We define

$$\theta(R) = \inf_{(f^1,B)\in R} \theta(f^1,B) \qquad (14)$$

θ(R) satisfies 1) for any θ<θ(R), R(θ) is empty; 2) for any θ>θ(R), R(θ) is non-empty. Therefore we have η=θ(R). The model parameters can be estimated at η:

$$(\hat{f}^1,\hat{B}) = \arg\inf_{(f^1,B)\in R} \max_m \frac{\left|\hat{f}^m - f^m(f^1,B)\right|}{\Delta^m} \qquad (15a)$$

This is a minimal-maximum problem. For the stiff string model this becomes

$$(\hat{F},\hat{G}) = \arg\inf_{(F,G)\in R} \max_m \frac{\left|\hat{f}^m - m\sqrt{F+(m^2-1)G}\right|}{\Delta^m} \qquad (15b)$$

We can then calculate f^1 and B by the inverse mapping of (4)

$$\hat{f}^1 = \sqrt{\hat{F}},\ \hat{B} = \hat{G}/\hat{F} \qquad (15c)$$

We call $e^m(F,G) = \frac{\left|\hat{f}^m - f^m(F,G)\right|}{\Delta^m}$ the relative frequency estimation error. Equations (15a) show that by shrinking R to zero-size, we locate the parameter pair that minimizes the maximal relative frequency error of all the given estimates.

The implementation of the minimal-maximum search greatly benefits from the fact that the gradient ∇e^m has constant direction. Using this property we can show that if (F,G)∈R is a local minimal-maximum of e^m, then it is also the minimal-maximum in R. In other words, the minimal-maximum of e^m is unique. A key proposition for finding the minimal-maximum is given below.

Proposition 1: if (F0, G0)∈R is not a minimal maximum, and $e_1 = e_2 = \ldots = e_K$ are K equalling maxima at (F0, G0), K>2, then there exist $l1$ and $l2$, 1≤$l1$, $l2$≤K, so that ∀1≤k≤K, along the decreasing direction of $e_{l1} = e_{l2}$, e_k - e_{l1} is non-increasing.

Proposition 1 ensures that we can always search down a curve $e_{l1} = e_{l2}$ without losing track of the maximum. The search come to a stop when there is another $l3$ so that $e_{l1} = e_{l2} = e_{l3}$. If this is not the minimal-maximum, we continue the search on curve $e_{l1} = e_{l3}$ or $e_{l2} = e_{l3}$, in the decreasing direction. It can be shown that the minimal-maximum can be reached in finite number of steps.

2.6. Detection in the presence of other harmonic particles

In polyphonic music we have multiple concurrent pitched events. It is usual that we have multiple harmonic particles in the same frame. In [7] the detection of multiple pitches is addressed as iteratively detecting and removing pitched events. In harmonic sinusoid modeling we can also detect multiple harmonic particles in a similar iterative way. Instead of removing detected events, we ignore the spectral peaks that are already collected in other harmonic particles. The harmonic grouping process remains the same. We are able to ignore certain peaks by virtue that unfound partials do not critically affect harmonic grouping. However, this makes it easier for spurious peaks to be collected. We split the harmonic grouping in two stages. In the first stage, we skip a partial whenever there is an already used peak in its searching range; in the second stage, with R already reduced to a small size, we review these skipped partials. If the used peak is still within the searching range, and it is the only peak within the range, then it is appointed to the new particle (as a *shared peak*). However, if there is another peak within the range, we take the following actions.

Let the partial index, frequency and amplitude estimates, and the f^1-B range of the used peak 1 be m_1, $\hat{f}_1$, $\hat{a}_1$ and R_1, of the unused peak 2 be m_2, $\hat{f}_2$, $\hat{a}_2$ and R_2. We define

$$s(\hat{a},\hat{f},m,R) = s_a(\hat{a}) + s_f^m(d(\hat{f},m,R), s_a(\hat{a})) \qquad (16)$$

and compare $s(\hat{a}_1,\hat{f}_1,m_1,R_1) + s(\hat{a}_2,\hat{f}_2,m_2,R_2)$ with $s(\hat{a}_1,\hat{f}_1,m_2,R_2) + s(\hat{a}_2,\hat{f}_2,m_1,R_1)$. If the former is larger, we collect peak 2 into the new harmonic particle; if the latter is larger, we replace peak 1 in the old harmonic particle with peak 2, and collect peak 1 into the new harmonic particle.

3. TRACKING HARMONIC PARTICLES

Let p_l be a harmonic particle at the l^{th} frame in time, with the f^1-B range R_l. Regarding f^1 and B of the l^{th} and $(l+1)^{th}$ frame, we assume that

$$\begin{cases} B_{l+1} = B_l \\ \left| f_{l+1}^1 - f_l^1 \right| < \Delta_l \end{cases} \tag{17}$$

(17) is the harmonic version of the frequency jump limiting used for sinusoid tracking [1]. The first line says that the inharmonicity feature remains constant during the same event, and the second line says that the pitch is not allowed to vary too fast. Given R_l and (17), we have the following inequality for the m^{th} partial at the $(l+1)^{th}$ frame:

$$f^m(f_-^1(R_l) - \Delta_l, B_l) < f_{l+1}^m < f^m(f_+^1(R_l) + \Delta_l, B_l) \tag{18}$$

This provides a range to look for any partial in the $(l+1)^{th}$ frame. To find a harmonic particle at frame $l+1$ as the successor of p_l, we initialize R_{l+1} by expanding R_l along the f^1 axis by Δ_l on both sides, i.e.

$$R_{l+1} = \{(f^1, B) \mid \exists \delta \in (-\Delta_l, \Delta_l), (f^1 + \delta, B) \in R_l\} \tag{19}$$

It can be shown that this expansion does not preserve the linearity of the sides of polygon R, so the R_{l+1} initialized strictly by (19) is no longer a polygon in the F-G plane. However, as an approximation, we can initialize R_{l+1} by expanding the vertices of R_l using (19) then take the convex hull (Figure 3). We can show that by taking this approximation R_l is expanded a little more than the amount in (19) near the sides, i.e. we are allowing a little more pitch variation at certain B's. This is not a big problem since Δ_l itself is not required to be very precise.

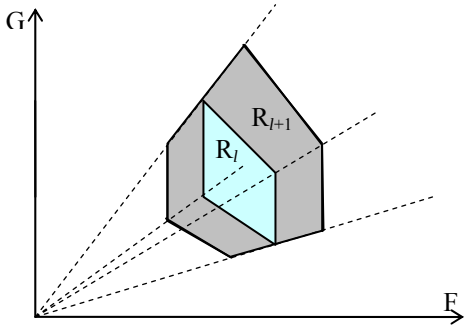


Figure 3: *Expanding R to allow pitch variation*

3.1. Short-term continuity

Once R_{l+1} has been initialized, the harmonic particle searching can be carried out using the method in section 2. However, with the knowledge of the predecessor particle, we are able to include short-term continuity criteria in the harmonic grouping stage by comparing the current candidates to the previous harmonic particle.

3.1.1. Frequency continuity

The frequency continuity has already been used to initialize R_{l+1}. However, we may have competing pitches within the allowed pitch jump. This is comparable to completing peaks in standard sinusoid modeling. In sinusoid modeling the successor is often chosen to be the peak with the smallest frequency jump [1,2] or the peak that gives the smoothest frequency contour [8, 9]. Similarly, in case of competing pitches, we choose to favour small pitch jumps. This is implemented using a pitch continuity score

$$s_p(l, l+1) = 1 - \left| \frac{f_{l+1}^1 - f_l^1}{\Delta_l} \right|^p, \; p \geq 1 \tag{20}$$

The exponent p tunes the balance of small and large pitch variations. The less is p, the less we tolerate large pitch jumps.

3.1.2. Amplitude continuity

Short-term amplitude continuity compares the partial amplitudes of candidate harmonic sinusoids of frame $l+1$ to those of frame l. We measure the similarity of two amplitude vectors $\{\hat{a}_l^m\}_{m=1,2,\cdots}$ and $\{\hat{a}_{l+1}^m\}_{m=1,2,\cdots}$ with

$$s_a(l, l+1) = \frac{2\sum_m \hat{a}_l^m \hat{a}_{l+1}^m}{\sum_m (\hat{a}_l^m)^2 + \sum_m (\hat{a}_{l+1}^m)^2} = (s_a)_1 (s_a)_2 \tag{21a}$$

$$(s_a)_1 = \frac{2\sqrt{\sum_m (\hat{a}_l^m)^2}\sqrt{\sum_m (\hat{a}_{l+1}^m)^2}}{\sum_m (\hat{a}_l^m)^2 + \sum_m (\hat{a}_{l+1}^m)^2}, (s_a)_2 = \frac{\sum_m \hat{a}_l^m \hat{a}_{l+1}^m}{\sqrt{\sum_m (\hat{a}_l^m)^2}\sqrt{\sum_m (\hat{a}_{l+1}^m)^2}} \tag{21b}$$

s_a combines two types of continuities: the total amplitude continuity $(\underline{s}_a)_1$, and the amplitude distribution continuity $(\underline{s}_a)_2$. $(\underline{s}_a)_1$ is a measure of the change in total volume; $(\underline{s}_a)_2$ is a measure of the change in short-time timbre. $0 \leq (\underline{s}_a)_1 \leq 1$, $0 \leq (\underline{s}_a)_1 \leq 1$.

However, we have observed that if the frequency has fast variation, the short-time amplitude continuity (21a) becomes questionable, especially when the event has a formant structure. This is due to the large variation of the short-time timbre that accompanies pitch changes. In this case we use a long-term amplitude continuity criterion, as follows.

3.2. Long-term amplitude continuity

Long-term amplitude continuity criterion is useful for events involving repetitive pitch variation. It assumes that the amplitude distributions of two frames on the same event are similar if its pitches in these two frames are close. Therefore instead of comparing the amplitude distribution with the frame closest in time, we compare it with the frame closest in frequency. Let the current harmonic sinusoid track contain frames 1, 2, …, l, with fundamental frequencies f_1^1, f_2^1, …, f_l^1, and let f_{l+1}^1 be a candidate fundamental frequency of the $(l+1)^{th}$ frame. We select $\bar{l}$ between 1 and l so that $f_{\bar{l}}^1$ is closest to f_{l+1}^1, i.e.

$$\bar{l} = \arg\min_{1 \leq k \leq l} \left| f_{l+1}^1 - f_k^1 \right|. \tag{22a}$$

We define the long-term amplitude distribution continuity score as

$$(s_a)_3 = \frac{\sum_m \hat{a}_{\bar{l}}^m \hat{a}_{l+1}^m}{\sqrt{\sum_m (\hat{a}_{\bar{l}}^m)^2}\sqrt{\sum_m (\hat{a}_{l+1}^m)^2}} \tag{22b}$$

(21a) can then be replaced by

$$s_a(l, l+1) = (s_a)_1 (s_a)_3 \tag{22c}$$

Although in (21a) and (22c) we are combining the two types of amplitude continuity measures by direct multiplication, it is not compulsory. We can use any other combination methods as long as $0 \le s_a \le 1$, with the identity $s_a=1$ for identical amplitude vectors.

3.3. Extending harmonic sinusoids

Let $p_{1,1}$, $p_{1,2}$, $p_{1,3}$, … be harmonic particles detected at frame *1*, and h_1, h_2, h_3, … be the harmonic sinusoids these particles are associated with. Now we look at the task of finding successor particles to $p_{1,1}$, $p_{1,2}$, $p_{1,3}$, …, so that h_1, h_2, h_3, … can be extended 1 frame forward. This task differs from the detection of concurrent harmonic particles (section 2.6) in that the particles detected in frame l+1 must satisfy additional continuity with the previous frame. Therefore, instead of using (12) to compare competing results, we use

$$s(l,l+1) = s_p(l,l+1) + s_a(l,l+1) \tag{23}$$

where the two addends are defined by (20) and (21a) (or (22c)). Like the detection of concurrent harmonic particles, the extension of concurrent harmonic sinusoids is performed in an iterative way, i.e. after the first harmonic particle is detected, additional harmonic particles are detected in the presence of the already found ones. The searching method remain the same, except for the scoring function (23) and the initialization of $R_{l+1,k}$ with $R_{l,k}$, k=1, 2, ….

If a successor for $p_{l,k}$ cannot be found at the $(l+1)^{th}$ frame, or any successor found for $p_{l,k}$ cannot meet a minimal continuity score, then h_k is terminated at frame l. This is the harmonic version of the *death* of a sinusoid track.

3.4. Forward harmonic sinusoid tracking

Forward harmonic sinusoid tracking creates, continues, and kills harmonic sinusoids in the forward procession of time. It takes pre-detected spectral peaks as input, and outputs harmonic sinusoids.

The forward harmonic particle tracking proceeds as follows. Let $p_{1,1}$, $p_{1,2}$, $p_{1,3}$, … be harmonic particles detected at frame 1. We associate each of them with a harmonic sinusoid, say h_1, h_2, h_3, …, i.e. $p_{1,k}$ is h_k constrained at the 1st frame. For l=2, 3, …, we do the following.

1) find the most continuous successors for the existing harmonic sinusoids (section 3.3);
   1.1) initialize $R_{l,1}$ with $R_{l-1,1}$, detect harmonic particle $p_{l,1}$;
   1.2) for k=2, 3, …, do 1.3;
      1.3) initialize $R_{l,k}$ with $R_{l-1,k}$, detect harmonic particle in the presence of $p_{l,1}$, $p_{l,2}$, …, $p_{l,k-1}$ using continuity score (23), or terminate h_k in case of failure;
2) find harmonic particles in the presence of the harmonic particles detected in 1), initialize a new harmonic sinusoid with each new harmonic particle.

3.5. Post-tracking parameter estimation

Pre-detected short-time sinusoid atoms are usually estimated using a stationary sinusoid assumption. However, accurate parameter estimation is possible only when the estimator considers parameter dynamics within a frame. Rather than estimating local dynamics from the spectrum, such as in [10], we access the dynamic information from the sinusoid tracks [11]. Post-estimation proceed in an iterative way. In each iteration, we do the following:

1) interpolate the frequency estimates using a cubic spline;
2) reestimate amplitudes using the interpolated frequency with

$$\hat{a}e^{i\hat{\varphi}} = \frac{\sum_{n=0}^{N-1} w_n^2 x_n e^{-j2\pi \int_{n/2}^{n} f(t)dt}}{\sum_{n=0}^{N-1} w_n^2} \tag{24}$$

where w_n ($0 \le n < N$) is a low-pass window function and x_n is the signal being analyzed;
3) interpolate the amplitudes using a cubic spline;
4) reestimate the frequencies by finding an approximate solution of

$$\hat{f} = \frac{\sum_{n=1}^{N-1}\sum_{m=0}^{n-1} w_{mn} a_n a_m \int_m^n f(t)dt \operatorname{sinc}\frac{\Delta\varphi_{mn}}{\pi}}{\sum_{n=1}^{N-1}\sum_{m=0}^{n-1} w_{mn} a_n a_m (n-m) \operatorname{sinc}\frac{\Delta\varphi_{mn}}{\pi}} \tag{25}$$

where $\Delta\varphi_{mn} = \int_m^n f(t)dt - 2\pi(n-m)\hat{f}$.

More details about (24) and (25) can be found in [11].

4. EXPERIMENTAL RESULTS

We run numerical tests on synthesized signals, for which the ground truth is available. The synthesized samples are 44100 points long. Amplitude and frequency variation rules include constant, exponential, and sinusoid-modulated variations. Stiff string model is applied to constant-frequency sounds. Partial amplitudes are designed to follow a 1/m rule, i.e. amplitudes are reciprocal to the partial index. We use the frame size 1024 and hop size 512. The fundamental frequency ranges from 5 bins to 40 bins (1bin=1/1024), spanning 3 octaves. We sample this range every semitone, i.e. at 37 different pitches. White noises are added to the test sampled optionally.

We measure two types of error: a harmonic grouping error and a waveform model error. The harmonic grouping error is measured by the number of correctly collected short-time sinusoid atoms divided by the total number of atoms. The waveform model error is measured by a signal-to-noise ratio, where the noise refers to the difference between the original source waveform and the resynthesized harmonic sinusoid waveform. The errors are measured independently for each test sample, which are then averaged over groups of samples.

4.1. Constant harmonic sinusoids

This group includes 925 test samples, with the 37 fundamental frequencies (f^1) from 5bins to 40 bins, 5 stiffness coefficients (B) from 0 to 0.0008, and 5 signal-to-noise ratios (SNR) from -15dB to 45dB. Given the three parameters, the test signal is synthesized by

$$M = \left\lfloor \frac{0.35}{f^1} \right\rfloor, \quad x_n = \sum_{m=1}^{M} \frac{1}{m} \cos(\varphi^m + 2\pi f^m(f^1, B)n) + r_n \tag{26}$$

The phase angles φ^m are taken at random. The noise r has been amplified to meet the selected SNR. The results are given Table 1. For stationary sinusoids the modeling is very successful, with more than 99.9% sinusoid peaks correctly collected into the partials when the SNR is above 15dB. We constantly get slightly better results for higher stiffness coefficients. This is due to the constraint of B above zero, which makes it easier to collect spurious peaks with a positive frequency departure than a negative one.

SNR / B	-15dB	0dB	15dB	30dB	45dB
0	27.5	86.4	99.9	100	100
0.0002	37.1	93.0	100	100	100
0.0004	40.6	94.2	100	100	100
0.0006	43.6	95.5	100	100	100
0.0008	44.9	95.8	99.9	100	100

(a) Group 1: peak collection rate (%)

SNR / B	-15dB	0dB	15dB	30dB	45dB
0	-0.9	14.8	30.6	45.7	60.7
0.0002	0.3	16.2	32.1	47.2	62.1
0.0004	0.6	16.5	32.4	47.5	62.3
0.0006	0.8	16.8	32.7	47.7	62.6
0.0008	1.0	17.0	32.8	47.9	62.7

(b) Group 1: Resynthesis SNR (dB)

Table 1: *Results for constant harmonic sinusoids.*

4.2. Constant pitch with exponential amplitude

Exponential amplitudes are found in real-world free vibrating bodies. This group includes 1850 test samples, with 37 fundamental frequencies (f^1) from 5 bins to 40bins, 2 stiffness coefficients (B) 0 and 0.0005, 5 amplitude decay rates (α) at -0.5, -1, -1.5, -2, -2.5 dB/frame (here "per frame" means per hop size, i.e. per 512 points) , and 5 SNRs from -15dB to 45dB. Given the four parameters, the test signal is synthesized as

$$x_n = \sum_{m=1}^{M} \frac{10^{\alpha n/10240}}{m} \cos(\varphi^m + 2\pi f^m(f^1, B)n) + r_n \qquad (27)$$

where M, φ^m and r are determined in the same way as in (26). The results are given in Table 2.

SNR / α	-15dB	0dB	15dB	30dB	45dB
-0.5	28.1	78.5	99.2	100	100
-1	21.5	55.4	84.5	98.9	100
-1.5	17.4	40.6	63.6	84.7	96.7
-2	14.8	32.7	49.6	67.7	81.6
-2.5	13.7	27.6	41.8	55.4	69.0

(a) Group 2: peak collection rate (%)
α: amplitude decay rate (dB/frame)

SNR / α	-15dB	0dB	15dB	30dB	45dB
-0.5	-0.2	15.2	31.0	46.3	61.2
-1	-0.5	13.9	30.0	45.6	59.7
-1.5	-0.9	12.9	21.7	44.5	56.1
-2	-1.3	11.8	23.7	43.2	49.3
-2.5	-1.8	12.5	21.9	26.7	22.0

(b) Group 2: Resynthesis SNR (dB)

Table 2: *Results for exponential amplitudes.*

The decay rate has a very regular effect on both errors, partially because the signal drops below noise level after certain points. Although in this test all partials have the same decay rate, for partial-dependent decay rates, which is common in real music signals, the behaviour is similar: all partials that falls below the noise level become hard to pick up. Unlike matching pursuits, sinusoid modeling does not assume any specific coupling between partial amplitudes.

4.3. Constant pitch with modulated amplitude

This group includes 550 samples, with 22 fundamental frequencies (f^1) from 5bins to 40bins (3 octaves on diatonic scale) , 5 modulation depths (d) 0.1, 0.2, …, 0.5, 5 modulator periods (T) 2, 4, …, 10 frames, SNR is fixed at 15dB. Given the four parameters, the test signal is synthesized as

$$x_n = \sum_{m=1}^{M} \frac{1}{m}\left(1 + d\cos\frac{\pi n}{256\mathrm{T}}\right)\cos(\varphi^m + 2\pi m f^1 n) + r_n \qquad (28)$$

where M, φ^m and r are determined in the same way as in (26). The results are given in Table 3.

d \T	all
all	> 99.98%

(a) Group 3: peak collection rate

d \ T	2	4	6	8	10
0.1	28.17	30.34	30.55	30.57	30.60
0.2	24.64	29.57	30.36	30.56	30.56
0.3	21.85	28.60	30.15	30.42	30.49
0.4	19.74	27.54	29.77	30.31	30.44
0.5	18.09	26.58	29.48	30.17	30.39

(b) Group 3: Resynthesis SNR (dB)
d: modulation depth; T: modulator period (frames)

Table 3: *Results for exponential amplitudes.*

With the SNR at 15dB, the partial collection rate stays consistently close to 100%. The waveform error increases with modulation depth and frequency.

4.4. Pitch modulation with constant amplitudes

This group includes 550 samples, with 22 fundamental frequencies (f^1) from 5bins to 40bins (3 octaves on diatonic scale), 5 modulator amplitudes (d) 0.3, 0.6, …, 1.5 semitones, 5 modulator periods (T) 2, 4, …, 10 frames, SNR ratio is set to 15dB. Given the four parameters, the test signal is synthesized as

$$x_n = \sum_{m=1}^{M} \frac{1}{m}\cos\left(\varphi^m + 2\pi m \sum_{l=0}^{n-1} f^1\left(1 + \left(2^{\frac{d}{12}} - 1\right)\cos\frac{\pi l}{256\mathrm{T}}\right)\right) + r_n \qquad (29)$$

where M, φ^m and r are determined in the same way as in (26). Again the partial collection rate stays consistently close to 100%. We list the resynthesis SNR's in Table 4. Only amplitude reestimation is used in the post-tracking stage to generate these results.

d \ T	2	4	6	8	10
0.3	14.5	23.6	27.9	29.0	29.3
0.6	10.8	17.9	21.5	25.4	27.0
0.9	7.7	14.7	17.7	21.3	24.0
1.2	6.0	11.2	13.0	18.5	21.0
1.5	4.8	8.3	7.8	13.0	18.9

Group4: Resynthesis SNR (dB)
d: modulator amplitude (semitones); T: modulator period (frames)

Table 4: *Results for vibrato.*

If we compare Table 4 with Table 3(b), we see that a frequency modulation of as small as 0.3 semitones brings more error than an amplitude modulation of 50% the central value.

5. CONCLUSIONS

In this article we have proposed a harmonic sinusoid modeling system, and discussed the harmonic sinusoid analyzer in brief. The harmonic sinusoid model is an update to the standard sinusoid model. Unlike sinusoid models that describe mostly low-level spectral contents, harmonic sinusoids directly model pitched events, which could provide solid starting points for music-related tasks. An application of this model in audio editors has been proposed in [12].

The current model can be further improved on several aspects. 1) The partial harmony has its origin in 1-dimension simple harmonic oscillation in string and air column, and does not describe membrane or bar vibration, which lies behind percussion instruments such as the kettledrum and marimba [13]. The analysis of these instruments requires partial frequency coupling rules different from simple harmony. 2) Even for harmonic instruments, there may exist extra partials that do not fall within a harmonic context [14]. These can be picked up by introducing individual spectral lines into the model, or be included in a more comprehensive harmonic model. 3) Harmonic tracking can be further refined by introducing finer frequency and amplitude continuity criteria, and the use of object models in partial tracking. 4) In [12] we have proposed the use of forward-backward searching [15] where atoms can be located at multiple frames, so that the tracking is more robust to local disturbance. 5) The current model treats very close (or overlapping) partials from two or more harmonic sinusoids as a shared partial; we also need separation techniques to resolve these shared partials into individual harmonic sinusoids. 6) On the synthesizer side, a more robust and accurate modeling of time-varying sinusoids is necessary to achieve better SNRs.

6. ACKNOWLEDGEMENTS

This work was supported by EPSRC EP/E017614/1 project OMRAS2 (Online Music Recognition and Searching) and Centre for Digital Music.

7. REFERENCES

[1] R. J. McAulay, T. F. Quatieri, "Speech analysis/synthesis based on a sinusoidal representation，" *IEEE Tran. ASSP*, vol. 34, pp. 744-754, 1986.

[2] X. Serra, Musical Signal Processing, chapter Musical Sound Modeling with Sinusoids plus Noise, pp. 91–122, G. D. Poli and A. Picialli and S. T. Pope and C. Roads Eds. Swets & Zeitlinger, 1996.

[3] R. Gribonval and E. Bacry, "Harmonic decomposition of audio signals with matching pursuit," *IEEE Tran. Signal Proc.*, vol.51 no.1, 2003.

[4] C. Duxbury, N. Chetry , M. Sandler and M. E. Davies , "An efficient two-stage implementation of harmonic matching pursuit," in *Proc. EUSIPCO04*, Vienna, 2004.

[5] F. Keiler, S. Marchand, "Survey on extraction of sinusoids in stationary sounds," in *Proc. DAFx'02*, 2002, pp.51-58.

[6] A. Klapuri, "Wide-band pitch estimation for natural sound sources with inharmonicities," in *Proc. AES 106th Convention*, Munchen, 1999.

[7] A. Klapuri, T. Virtanen, J.-M. Holm, "Robust multipitch estimation for the analysis and manipulation of polyphonic musical signals," in *Proc. DAFx'00*, 2000.

[8] M. Lagrange, S. Marchand, M. Raspaud, J.-B. Rault, "Enhanced partial tracking using linear prediction," in *Proc. DAFx'03*, 2003.

[9] P. Depalle, G. Garcia, X. Rodet, "Tracking of partials for additive sound synthesis using hidden Markov models," in *Proc. ICASSP*, Minneapolis, 1993.

[10] A. Röbel, "Estimating partial frequency and frequency slope using reassignment operators," in *Proc. ICMC*'02, Göteborg, 2002.

[11] Wen X., M. Sandler, "Error compensation in modeling time-varying sinusoids," in *Proc. DAFx'06*, Montreal, 2006.

[12] Wen X., M. Sandler, "New audio editor functionality using harmonic sinusoids," in *Proc. AES 122nd Convention*, Vienna, 2007.

[13] N. H. Fletcher and T. D. Rossing, The Physics of Musical Instruments (2nd Edition), Springer-Verlag New York, Inc., 1998.

[14] H. A. Conklin, "Generation of partials due to nonlinear mixing in a stringed instrument", *J. Acoust. Soc. Am.*, vol.105, no.1, January 1999, pp.536-545.

[15] S. Austin, R. Schwartz, P. Placeway, "The forward-backward search algorithm," in *Proc. ICASSP91*, Toronto, 1991.

OBJECT CODING OF HARMONIC SOUNDS USING SPARSE AND STRUCTURED REPRESENTATIONS

Grégory Cornuz[1], *Emmanuel Ravelli*[1,2], *Pierre Leveau*[1,2], *Laurent Daudet*[1]

[1]Institut Jean Le Rond d'Alembert, LAM team
Université Pierre et Marie Curie - Paris 6
11, rue de Lourmel
75 015 Paris - France
lastname@lam.jussieu.fr

[2]TSI department
GET-ENST (Télécom Paris)
37-39, rue Dareau
75 014 Paris - France
firstname.lastname@enst.fr

ABSTRACT

Object coding allows audio compression at extremely low bit-rates, provided that the objects are correctly modelled and identified. In this study, a codec has been implemented on the basis of a sparse decomposition of the signal with a dictionary of Instrument-Specific Harmonic atoms. The decomposition algorithm extracts "molecules" i.e. linear combinations of such atoms, considered as note-like objects. Thus, they can be coded efficiently using note-specific strategies. For signals containing only harmonic sounds, the obtained bitrates are very low, typically around 2 kbs, and informal listening tests against a standard sinusoidal coder show promising performances.

1. INTRODUCTION

Audio coding has traditionally evolved in two directions, depending on the target bitrate. At high rates, state-of-the-art audio coding is transform-based (e.g. MPEG4-AAC and MPEG4-TwinVQ [1]). At lower rates, parametric coders perform slightly better. MPEG4-SSC [2], based on a sinusoids+transients+noise model, outperforms MPEG4-AAC at 24kits; but is not designed for lower bitrates. MPEG4-HILN [3], based on a harmonics+sinusoids+noise model, works at lower bitrates but its performance appears to be very signal dependent and on average it is comparable to MPEG4-AAC at 16kbits and MPEG4-TwinVQ at 6kbits ; the benefit compared to the transform-based coders is that HILN allows additional functionality such as speed and pitch modifications at the synthesis.

For more flexibility on the type of possible transform-domain sound modifications (for instance the modification of timbre parameters of a single instrument in a polyphonic mixture), it is necessary to go one step further in the understanding of the contents of the audio file; this is the goal of so-called object-based audio coding, which fits well in the general context of MPEG4. Instead of coding transform coefficients or parameters of a low-level model, object audio coders consider higher-level "sound objects", consisting ideally of individual notes or chords. In [4], pitched sound objects consisting of the sum of harmonic sinusoidal partials are efficiently estimated using a statistical approach; the resulting coder appears to perform better than transform and parametric coders on solo or duo of harmonic instruments at 8kbit/s and 2 kbit/s. However, this approach requires extensive computational ressources which makes them unpractical for most applications.

In this paper, we present a novel object-based coding, which allows the computation of objects in a reasonable computational time. First, the sound is decomposed with a dictionary of instrument-dependent atoms, or groups of atoms ("molecules"), with a modified version of the matching pursuit algorithm. Then, the atom parameters are encoded with variable precision. The main benefit of this approach is that very low bit-rates can be achieved at full bandwidth, while keeping an acceptable sound quality for most sound examples. The price to pay, besides computational complexity, is the necessity to store the full database of atoms at both encoder and decoder, a requirement that is more and more acceptable given the increase in storage capacities. This paper is organized as follows : in section 2, we describe the decomposition process into sound objects. In section 3, we detail how we encode the extracted sound objects. Finally, preliminary results are given in section 4.

2. DECOMPOSITION ALGORITHM

2.1. Signal Model

2.1.1. Instrument Specific Harmonic Atoms

The signal is modelled as a linear combination of N harmonic atoms $h_{s_n,u_n,f_{0_n},c_{0_n},A_n,\Phi_n}$ parameterized in terms of scale s_n (atom duration), time localisation u_n, fundamental frequency f_{0_n}, fundamental chirp rate c_{0_n}, partial amplitudes $A_n = \{a_{m,n}\}_{m=1:M}$ and partial phases $\Phi_n = \{\phi_{m,n}\}_{m=1:M}$:

$$x(t) = \sum_{n=1}^{N} \alpha_n \, h_{s_n,u_n,f_{0_n},c_{0_n},A_n,\Phi_n}(t). \qquad (1)$$

Each harmonic atom can be written as

$$h_{s,u,f_0,c_0,A,\Phi}(t) = \sum_{m=1}^{M} a_m \, e^{j\phi_m} \, g_{s,u,m.f_0,m.c_0}(t). \qquad (2)$$

The amplitudes of the M partials are constrained to $\sum_{m=1}^{M} a_m^2 = 1$ and the signal corresponding to each partial is given by a *Gabor* atom normalized to unit energy

$$g_{s,u,f,c}(t) = w\left(\frac{t-u}{s}\right) e^{2j\pi\left(ft+\frac{c}{2}t^2\right)} \qquad (3)$$

with w as a weighting window.

When partial amplitudes are learned from a database (see 2.3.1), these atoms are called Instrument Specific Harmonic (ISH) atoms. Each amplitude vector A is then associated with a class i (in our case an instrument) and a discrete pitch value p, and is thus defined as belonging to a set $\mathcal{C}_{ip}$. Generally, several vectors are used for each class and each pitch value.

2.1.2. Instrument Specific Harmonic Molecules

The long-term structures, such as music notes, cannot be efficiently modelled with a single ISH atom. However, building sets of ISH atoms (named *molecules*) can overcome this issue. The constraints for atoms to belong to a single molecule are the following:

- the atoms span a range of time locations u, with exactly one atom per location,
- all atoms come from the same instrument,
- the log-variation of fundamental frequency between any two consecutive atoms is bounded by a threshold D:

$$|\Delta \log f_0| \leq D \tag{4}$$

2.2. Decomposition algorithms

2.2.1. The Matching Pursuit Algorithm

Given a ISH dictionary, the problem becomes that of decomposing the signal as a collection of molecules of ISH atoms from this dictionary. A popular and efficient method to achieve atomic decompositions is the Matching Pursuit algorithm [5]. It can be modified for molecular decompositions [6, 7]. The Matching Pursuit algorithm proceeds as follows:

1. The correlations between the signal and all the atoms h of the dictionary are computed using inner products $\langle x, h\rangle = \sum_{t=1}^{T} x(t)\, h(t)$.
2. The atom h that has the largest absolute correlation $|\langle x, h\rangle|$ with the signal is selected, then subtracted from the signal with a weighting coefficient $\alpha = \langle x, h\rangle$.
3. Correlations are updated on the residual signal, and the algorithm is iterated to step 2 until the stopping condition is satisfied. This condition can be a target Signal-to-Residual energy Ratio (SRR), or a fixed number of iterations.

2.2.2. Molecular Algorithm

The algorithm that is here briefly introduced is fully described in [8]. Its flowchart is presented on Figure 1. Its main feature is to iteratively extract molecules of ISH atoms using a Matching Pursuit algorithm that has been modified as follows :

- **Best atom path selection:** an atom path is selected in instrument-specific time-pitch planes using dynamic programming. The search zone of this path is delimited around a *seed* atom: the atom that is the most correlated with signal. A threshold on the atom weights α_n is set to avoid the selection of low-amplitude atoms, and as a consequence to reduce the amount of data to encode.
- **Atom parameters tuning on the path**: the fundamental chirp rate c_0 is estimated jointly with the refinement of the fundamental frequency f_0 using a maximization of the inner product $|\langle x, h\rangle|$ with regard to f_0 and c_0. The partial phases ϕ_m of each atom of the molecule are computed using the following formula:

$$e^{j\phi_m} = \frac{\langle x, g_{s,u,m.f_0,m.c_0}\rangle}{|\langle x, g_{s,u,m.f_0,m.c_0}\rangle|}. \tag{5}$$

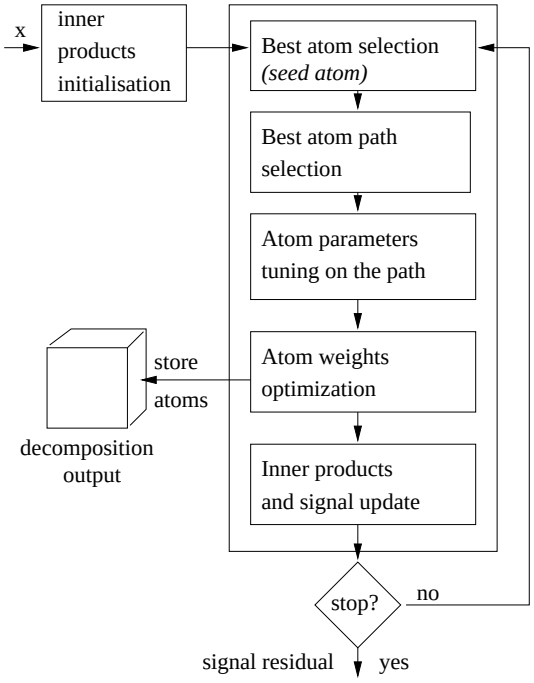


Figure 1: Flow chart of the algorithm for decomposing a signal into molecules of ISH atoms

- **Atom weights optimization**: the respective weights of each atom are re-estimated using an orthogonal projection of the signal on the subspace corresponding to the atoms of the molecule.

2.3. Sampling the dictionary

In practical applications, the search step can only be performed on a finite number of atoms. Thus, one has to sample the dictionary by making the atom parameters s, u and f_0 discrete:

- The scale s often spans a small set of powers of 2.
- The time localisation u is typically set to equally spaced time bins, with a time shift Δu set to a fraction of the atom scale.
- The fundamental frequency f_0 is sampled logarithmically. This is a noticeable difference with the Harmonic MP algorithm [6], where fundamental frequencies are sampled linearly.

The amplitude vectors A are already a discrete set of vectors and the phase vectors are estimated using Equation 5.

2.3.1. Learning the model

For the following experiments, the vectors of partial amplitudes $\{A_{i,p,k}\}_{k=1\ldots K}$ are learned for each instrument/pitch class $\mathcal{C}_{i,p}$ on isolated notes from three databases: the RWC Musical Instrument Sound Database [9], IRCAM Studio On Line [10] and the University of Iowa Musical Instrument Samples [11]. We select five instruments producing harmonic notes: oboe (Ob), clarinet (Cl), cello (Co), violin (Vl) and flute (Fl).

For each isolated note signal, the time frame with maximal energy is computed and all the subsequent time frames whose energy lies within a certain threshold of this maximum are selected. This relative threshold is set to a ratio of 0.05 in the following. The partial amplitudes are computed on each of these training frames by

$$a_m = \frac{|\langle x, g_{s,u,m\times f_0, m\times c_0}\rangle|}{\left(\sum_{m'=1}^{M} |\langle x, g_{s,u,m'\times f_0, m'\times c_0}\rangle|^2\right)^{1/2}} \qquad (6)$$

where f_0 and c_0 are tuned in order to maximize the SRR on this frame, using the same optimisation method as in the parameter tuning step. The vector of amplitudes is then associated to the pitch class p that is the closest to f_0. The resulting number of vectors per instrument and per pitch class are indicated in Table 1.

Inst.	N_{train}	N_{train} per pitch
Ob	5912	169
Cl	9048	193
Co	13868	285
Vl	37749	700
Fl	13216	330

Table 1: Total number of training time frames per instrument and average number per pitch class.

The size of the dictionary varies linearly as a function of the number of amplitude vectors. Since the number of vectors is too large to ensure computationally tractable decompositions, we chose to reduce the number of vectors by vector quantization: K amplitude vectors are kept for each class $\mathcal{C}_{i,p}$ using the k-means algorithm with the Euclidean distance.

3. PARAMETERS CODING

We use a simple scheme where a representation is first estimated from the signal (see previous section) and then the representation parameters are quantized and coded a posteriori. At the decoder, the quantized parameters are decoded and used to synthesize a new signal.

Two properties of the representation allow efficient coding at very low bit rate. Firstly, the molecular algorithm builds "objects", composed of a succession of atoms. The parameters of the atoms which belongs to the same molecule are highly correlated and thus can be efficiently coded. Secondly, due to the greedy nature of the molecular algorithm some parameters are already quantized before the coding stage; these parameters are consequently coded without any loss using entropy coding.

In the following, we list all the parameters of the model and the method we have chosen to code them.

- The scale s_n is constant and thus is not coded.
- The time localisation u_n is on a grid with a step size $s_n/2$. Only the absolute position of the first atom of a molecule is coded, the positions of the following atoms are then the consecutive values on the grid. The only additional parameter required by the decoder is the number of atoms that belong to the molecule.
- The fundamental frequency f_{0_n} of every atom is coded in its crude version (before the atom parameters tuning stage, see previous section). For the atoms of a molecule except the first one, we compute differences between consecutive values of the fundamental frequency, and the resulting values are entropy coded.
- The weight α of the first atom of a molecule is coded using a standard uniform quantizer + entropy coding approach [12]. The weights of the next atoms are coded using differential coding and uniform quantization.
- The partial amplitudes vectors A_n are already vector quantized. We then simply transmit the index of the corresponding vector in the dictionary. The index is composed by: the pitch class (crude version of the fundamental frequency, already coded) + the instrument class (coded one time for each molecule) + the index in the table of the corresponding pitch/instrument class. The index is entropy coded.
- We do not code the fundamental chirp rate c_{0_n} as we found that this parameter is not perceptually relevant enough given the necessary bit budget needed to code it.
- The phases are not coded. We use an alternative approach where the phases are interpolated at the decoder to ensure a continuity between the partials of the consecutive atoms.

4. EXPERIMENTS

The coder is evaluated on 5 solos (clarinet, cello, flute, oboe, violin) and 4 duos (clarinet/flute, cello/flute, cello/violin, flute/flute), extracted from commercial CDs (hence having no relationship with the single notes database used for learning).

The two steps of the coding process, namely the signal decomposition and the parameters coding, have been performed with the following parameters:

- **Sampling parameters:** for our application, the choice of a single scale s corresponding to a duration of about 50 ms is sufficient. It is long enough to have a good frequency resolution. Concerning the localization period Δu, it is here set to half the scale, short enough to track the perceptually relevant amplitude and frequency modulations of the signal that correspond to expressive features such as vibrato or tremolo (between 4 and 10 Hz). The fundamental frequency is sampled with a step of $1/10$ tone.
- **Decomposition parameters:** The general threshold for the decomposition has been set to 15 dB or 250 atoms per second. For the atom path formation, the difference between consecutive fundamental frequencies is the corresponding sampling step of the f_0 sampling: $1/10$ ton.
- **Quantization parameters:** The weight of the first atom of a molecule is quantized on 6 bits, and the weights of the next atoms are quantized on 4 bits. The order of the DPCM quantizer is set to one. The entropy coder we use for all parameters is the adaptive arithmetic coder from Witten et al. [13].

With these parameters, we obtain computation times equivalent to 10x real-time on a 3Ghz computer and Matlab, largely dominated by the decomposition algorithm.

4.1. Full codec and reduced codec

During the analysis stage, at the end of the decomposition, the molecular algorithm tends to produce molecules that do not correspond to underlying music notes in the performance. These

molecules of low energy are thus not perceptually nor physically relevant and are only extracted to reduce the overall SNR. As a consequence, the decomposition should be stopped before the apparition of such molecules. However, it is hard to find an analytical solution for a stopping criteria in the decomposition. Instead, in the framework of this study, we have prefered to manually decide the optimum number of iterations for each audio signal. A Matlab Graphical User Interface has thus been implemented (Fig. 2) where the user can listen to the synthesized signal in function of the number of iterations and thus choose the optimum number of iterations in the molecular algorithm. Such optima have been found, except for two files (Cello solo and Cello/Violin duo) where the original stopping criteria of the molecular algorithm gave the best results. We call the coder based on this manipulation the "reduced codec"; while the coder which encodes the complete set of molecules is the "full codec".

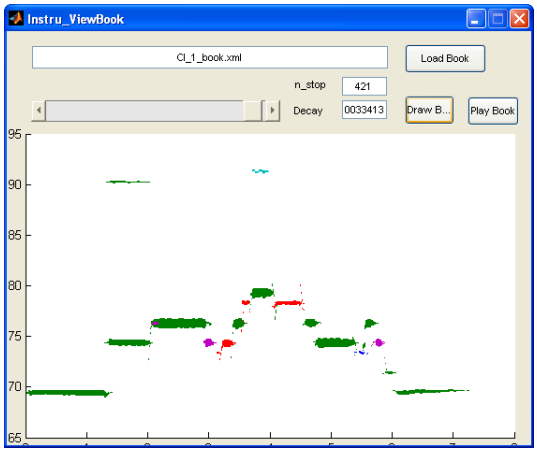


Figure 2: *Matlab Graphical User Interface allowing the user to visualize the representation and to select the optimal threshold for the decomposition. Different colors indicate different instruments labels.*

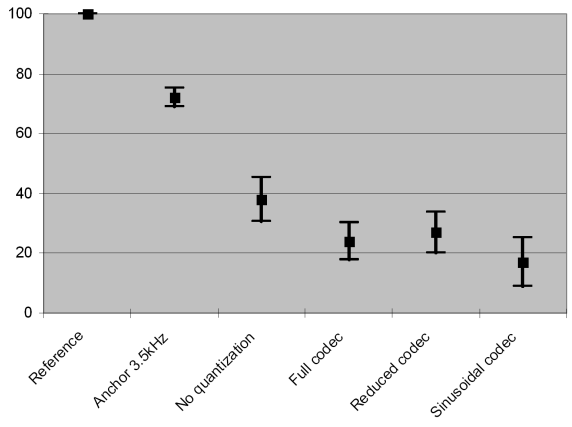


Figure 3: *MUSHRA overall mean scores*

	Full Codec (FC)	Reduced Codec (RC)
Clarinet	1.3	1.1
Cello	3.8	3.8 (*)
Flute	1.3	1.1
Oboe	2.6	1.0
Violin	4.4	2.3
Cl. / Fl.	2.4	1.9
Ce. / Fl.	1.6	1.5
Ce. / Vl.	3.9	3.9 (*)
Fl. / Fl.	4.6	2.6

Table 2: Bitrates (in kb per second) for each test file and the two variants of our codec. for the 2 files marked with an asterisk (*), the reduced codec was found equal to the full codec.

	HR	AN	NQ	FC	RC	SC
Clarinet	100	69	44	29	21	29
Cello	100	81	32	20	20	6
Flute	100	76	29	31	34	30
Oboe	100	70	40	12	20	18
Violin	100	66	62	33	33	14
Cl. / Fl.	100	74	41	36	42	36
Ce. / Fl.	100	74	25	15	21	8
Ce. / Vl.	100	70	43	13	13	6
Fl. / Fl.	100	68	30	23	21	3

Table 3: Mean of the MUSHRA scores for each version of each test signal

4.2. Listening tests

To evaluate our codecs, we performed several listening tests based on the standard MUSHRA method [14]. 15 persons took part in the listening tests to compare 5 versions of each signal: a hidden reference (HR), an anchor signal (AN) (3,5 kHz low-pass), the synthesized signal (NQ) obtained at the end of the molecular algorithm (without any quantization), the full codec (FC), the reduced codec (RC), and a simple frame-based sinusoidal coder used as a reference parametric codec (SC). The average bitrate of the codecs is around 3 kb per second for the Full Codec, and around 2 kb per second for the reduced codec (see Table 2). For the Sinusoidal Codec (SC), the bitrate was fixed at 2 kb per second. The mean of the scores obtained for each version of each signal are in Table 3. The overall means are in Fig. 3. These results first show that the reduced codec has performances that are similar or better than the full codec except for two files (Clarinet and Fl. / FL.), a case where more bits actually decrease the quality. It also shows that the reduced codec performs similarly or better than the reference sinusoidal coder except for one file (Clarinet).

5. CONCLUSION

In this paper we have described preliminary experiments that demonstrate that object-based coding of simple polyphonic music is both technically feasible and computationally tractable, when some hypothesis on the sources are verified (in our case, this is mainly an hypothesis of harmonicity). The resulting representations can achieve coding at bitrates as low as 2 kbs for monophonic sounds, with a sound quality that is in general comparable to sinusoidal coding, and in some cases significantly better. Further improve-

ments will be focused on three directions : first, we can improve on our decomposition model, for instance on estimating jointly the combinations of notes that optimally explains the signal for polyphonic music. Then, we can improve the quantization and coding techniques, in order to reduce the loss of sound quality due to quantization. Finally, we want to investigate criteria for stopping strategies in the Matching Pursuit decomposition process corresponding to the optimal codec (here called the reduced codec).

Another question, which is still open at the moment, is whether these techniques would still perform well with an increase of the number of instruments. It would as well be interesting to evaluate its performance on sounds that are not included in the training set but still verify the harmonicity assumption (for instance, voice). However, other classes of sounds such as percussive sounds or noisy sounds must be analyzed with different dictionaries, but object-based coding would still be relevant in this case.

6. ACKNOWLEDGEMENTS

The authors wish to thank Emmanuel Vincent for providing the reference sinusoidal coder, and the members of the LAM for taking part in the listening tests.

7. REFERENCES

[1] International Organization for Standardization, "ISO/IEC 14496-3:2001, information technology - coding of audio-visual objects - part 3: Audio," 2001.

[2] A.C. den Brinker, E.G.P. Schuijers, and A.W.J. Oomen, "Parametric coding for high-quality audio," in *Proceedings of the 112th AES Convention*, Munich, Germany, May 2002.

[3] H. Purnhagen and N. Meine, "HILN-the MPEG-4 parametric audio coding tools," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS 2000)*, May 2000, vol. 3, pp. 201–204.

[4] E Vincent and MD Plumbley, "Low bitrate object coding of musical audio using bayesian harmonic models," *To appear in IEEE Trans. on Audio, Speech and Language Processing*.

[5] S. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Trans. on Signal Processing*, vol. 41, no. 12, pp. 3397–3415, December 1993.

[6] R. Gribonval and E. Bacry, "Harmonic decomposition of audio signals with matching pursuit," *IEEE Trans. on Signal Processing*, vol. 51, no. 1, pp. 101–111, January 2003.

[7] L. Daudet, "Sparse and structured decompositions of signals with the molecular matching pursuit," *IEEE Trans. on Speech, Audio and Language Processing*, vol. 14, no. 5, pp. 1808–1816, September 2006.

[8] P. Leveau, E. Vincent, G. Richard, and L. Daudet, "Instrument-specific harmonic atoms for mid-level music representation," *to appear in IEEE Trans. on Speech, Audio and Language Processing*.

[9] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, "RWC Musical Instrument Sound Database," Distributed online at http://staff.aist.go.jp/m.goto/RWC-MDB/.

[10] "University of iowa music instrument samples database, http://theremin.music.uiowa.edu/mis.html," .

[11] "Ircam studio online database, http://forumnet.ircam.fr/402.html?l=1," .

[12] Allen Gersho and Robert M. Gray, *Vector Quantization and Signal Compression*, Springer, 1991.

[13] Ian H. Witten, Radford M. Neal, and John G. Cleary, "Arithmetic coding for data compression," *Commun. ACM*, vol. 30, no. 6, pp. 520–540, 1987.

[14] ITU, "ITU-R BS.1534-1: Method for the subjective assessment of intermediate quality levels of coding systems," 2003.

ADAPTIVE THRESHOLD DETERMINATION FOR SPECTRAL PEAK CLASSIFICATION

Miroslav Zivanovic
Universidad Pública de Navarra
Pamplona, Spain
miro@unavarra.es

Axel Roebel and Xavier Rodet
IRCAM
Paris, France
roebel@ircam.fr,
rod@ircam.fr

ABSTRACT

A new approach to adaptive threshold selection for classification of peaks of audio spectra is presented. We here extend the previous work on classification of sinusoidal and noise peaks based on a set of spectral peak descriptors in a twofold way: on one hand we propose a compact sinusoidal model where all the modulation parameters are defined with respect to the analysis window. This fact is of great importance as we recall that the STFT spectra are closely related to the analysis window properties. On the other hand, we design a threshold selection algorithm that allows us to control the decision thresholds in an intuitive manner. The decision thresholds calculated from the relationships established between the noise power in the signal and the distributions of sinusoidal peaks assures that all peaks described as sinusoidal will be correctly classified. We also show that the threshold selection algorithm can be used for different types of analysis windows with only a slight parameter readjustment.

1. INTRODUCTION

The decomposition of audio spectra in sinusoids, transients and noise is a useful tool for improving the results of parameter estimation and/or signal manipulation applications. As has been shown for the case of transient detection [1] and sinusoidal and noise discrimination [2], the classification of spectral peaks is a beneficial approach to identify signal components. Such a classification scheme that makes optimal use of the information provided by spectral peaks can then be used to achieve a robust segmentation into higher level signal components, e.g. partials or unvoiced regions.

The basis for spectral peak classification is an adequate choice of criteria that would best describe sinusoidal and noise spectral peaks of audio signals. Ideally, those criteria (from now on descriptors) would be able to precisely detect the nature of each peak in the spectrum and thus provide for a complete separation between the corresponding peak classes in the descriptor domains. Consequently, the decision boundary for the classification process would be unambiguous and no misclassfication of spectral peaks would occur. This scenario, however, is purely hypothetical as the peaks corresponding to sinusoids (partials) in the spectra of real-world signals are usually subject to additive noise and some type of modulation. In these cases the descriptor distributions of the different peak classes overlap and the optimal determination of the decision boundaries will depend on the specific application.

The peak classification method proposed in [2] makes use of descriptors that were designed to adequately characterize non-stationary sinusoidal signals. These descriptors have proven to lead to superior classification performance than other approaches devoted to sinudoidal detection/estimation [3,4]. It was shown in [2] that the peak classes can be characterized by distributions in the descriptor domains, similar to probability density functions. Once the distributions have been generated, a simple decision tree can be derived that allows the classification of spectral peaks into sinusoids, noise and sidelobes.

The peak classification method has been used successfully in a number of applications. As examples we mention polyphonic F0 detection [5], adaptive noise floor determination [6] and voiced unvoiced frequency boundary determination. Another interesting application would be the pre-selection of the sinusoidal peaks to reduce the number of candidate peaks considered for partial tracking in additive analysis. A reliable classification of noise peaks could reduce the number of incorrect connections and for probabilistic approaches like [7] it would considerably reduce the computational cost. The major problem with the classification scheme in [2] is the control of the classification boundaries (classification thresholds) that generally need adaptation for the specific problem at hand. A further problem is that the descriptor boundaries of the different classes will depend on the analysis window that is used. Up to know there did not exist a high level control parameter that would allow to adjust the sensitivity of the algorithm in an intuitive manner. There are two signal parameters that directly affect the classification boundaries. The first is the maximum modulation depth and period of the sinusoids. The second is the minimum amplitude of the sinusoids above the noise floor. Both parameters influence the boundaries of the sinusoidal class and accordingly both can be used to control the decision boundaries. The problem using the modulation limits as control parameter is the fact that the modulation is not a single parameter but a parameter vector of at least 4 dimensions (period and depth for amplitude and frequency modulation). Therefore, it can not be used to provide and intuitive control of the classification boundaries. On the other hand the sinusoidal peak amplitude above the noise floor is a single parameter that for a given modulation limit would allow us to control the complex decision thresholds rather intuitively.

Accordingly, in this paper we investigate into the relation between the peak amplitude above the noise floor and the descriptor boundaries for the class of sinusoidal peaks. The descriptors are defined and their properties discussed thoroughly in [2] but for sake of clarity we will give a brief resume of the most prominent characteristics in the section 2. For the sinusoidal model described in section 3 we define the space of sinusoidal components by selecting particular limits of the amplitude and frequency modulation rate and depths, as well as the modulation laws. In section 4 we present the descriptor distributions for the different signal classes and in section 5 we establish the mathematical model for the descriptor limits of the sinusoidal class as a function of the peak amplitude level above the

noise floor. In the experimental part in section 6 we show that the threshold model successfully adapts to the limits of the distributions of sinusoidal peaks for different types of analysis windows.

2. SPECTRAL PEAK DESCRIPTORS - SUMMARY

Being an elementary classification object, we define a spectral peak as the normalized energy spectral density between two contiguous minima in the DFT modulus $|X(k)|$ of the signal $x(n)$ multiplied by the analysis window. The spectral peak descriptors proposed in [2] are the Normalized Bandwidth Descriptor, the Normalized Duration Descriptor and the Frequency Coherence Descriptor. The first two are well suited to distinguish between sinusoidal and noise peaks while the third can be used to detect the sidelobe structure that is an artifact of the windowing process.

2.1. Normalized Bandwidth Descriptor (NBD)

Energy distribution along the frequency grid provides useful information for identifying the nature of the signal related to a given spectral peak. Being $X(k)$ the DFT of the windowed signal and considering L to be the number of samples in the spectral peak, we have defined the NBD as a function of mean frequency $\bar{k}$ and root mean square bandwidth BW_{rms}:

$$\mathrm{NBD} = \frac{BW_{rms}}{L} = \frac{1}{L}\sqrt{\frac{\sum_k \left(k-\bar{k}\right)^2 |X(k)|^2}{\sum_k |X(k)|^2}}, \quad (1)$$

$$\bar{k} = \frac{\sum_k k|X(k)|^2}{\sum_k |X(k)|^2}. \quad (2)$$

The sums are performed over the L bins in the peak under consideration.

2.2. Normalized Duration Descriptor (NDD)

As with mean frequency and bandwidth, the mean time and root mean square duration give a rough idea of the distribution of the signal related to a spectral peak along the time grid. The time duration for continuous signals has been defined in [8] as the standard deviation of the time with respect to the mean time. For discrete signals, the following expressions characterize the duration T_{rms} and mean time $\bar{n}$ respectively:

$$T_{rms} = \sqrt{\sum_n (n-\bar{n})^2 |x(n)|^2}, \quad (3)$$

$$\bar{n} = \sum_n n|x(n)|^2, \quad (4)$$

where $|x(n)|^2$ is the normalized signal's energy. It was shown in [8] that, from the duality of the Fourier transform, both mean time and duration can be expressed in terms of the spectrum. This important feature permits us to describe individual spectral peaks through the parameters generally employed in the time domain. Considering M to be the size of the analysis window, for discrete spectra the NDD can be obtained by means of:

$$\mathrm{NDD} = \frac{T_{rms}}{M} = \frac{1}{M}\sqrt{\frac{\sum_k \left(A'(k)^2 + (g_d(k)+\bar{n})^2\right)|X(k)|^2}{\sum_k |X(k)|^2}}, \quad (5)$$

$$\bar{n} = -\frac{\sum_k g_d(k)|X(k)|^2}{\sum_k |X(k)|^2}, \quad (6)$$

where $g_d(k)$ is the group delay and $A'(k)$ is the frequency derivative of the continuous magnitude spectrum. The group delay $g_d(k)$ is defined to be the derivative of the phase spectrum with respect to frequency. For a single bin of the DFT spectrum it equals the mean time according to [8] and specifies the contribution of this frequency to the center of gravity of the signal related to the spectral peak. This property of the group delay has been used in [9] to derive the time reassignment operator, which together with the frequency reassignment aims to improve signal localization in the time-frequency plane. According to [9] the group delay can be calculated efficiently by:

$$g_d(k) = -\mathrm{real}\frac{X_t(k)X^*(k)}{|X(k)|^2}, \quad (7)$$

being $X_t(k)$ the DFT of the signal using a time weighted analysis window. It can be shown that $A'(k)$ is the imaginary counterpart of the group delay in (7):

$$g_d(k) = -\mathrm{imag}\frac{X_t(k)X^*(k)}{|X(k)|^2}. \quad (8)$$

As for the NBD all the summations are done over all the bins in the spectral peak.

2.3. Frequency Coherence Descriptor (FCD)

The frequency reassignment operator for constant amplitude chirp signals points exactly onto the frequency trajectory of the chirp at the position of the centre of gravity of the windowed signal. The frequency offset Δ_ω between the frequency at the center of a DFT bin and the reassigned frequency in radians is given by:

$$\Delta_w(k) = \mathrm{imag}\frac{X_{dt}(k)X^*(k)}{|X(k)|^2}, \quad (9)$$

where $X_{dt}(k)$ is the DFT of the signal windowed by the time derivative of the analysis window. The Frequency Coherence Descriptor is defined as a minimum absolute frequency offset $\Delta_\omega(k)$ for all the bins belonging to that peak:

$$\mathrm{FCD} = \frac{N}{2\pi}\min_k |\Delta_w(k)|, \quad (10)$$

being N the number of bins in the DFT. The normalization factor in (10) ensures that the descriptor is expressed in bins of DFT.

3. SINUSOIDAL MODEL AND PEAK DISTRIBUTIONS

To be able to classify a sinusoidal component we need to define what we consider to belong to the sinusoidal class. As is common for sinusoidal modeling we are going to understand a sinusoidal component as a sinusoid with slowly varying amplitude and frequency parameters [10]. For an investigation into the properties of the spectral peak classes this requirement is not sufficient. To completely define the space of sinusoidal components we have to select concrete limits of the amplitude and frequency modulation rate and depths, and we have to specify a concrete form of the modulation laws.

For the present application there exists an obvious constraint for the modulation which is related to the fact that the spectrum of the sinusoidal component has to contain a dominant mainlobe. Otherwise the investigation of an individual spectral peak can not provide us with sufficient information about the underlying sinusoid. Accordingly, the modulation rate and depth have to be limited such that a dominant mainlobe is present in the Fourier spectrum of each sinusoidal component. Because frequency and time resolution are related to the window size and form, the modulation limits will depend on these two variables. A simple solution to ensure the modulation constraint described above for all window sizes is to determine the maximum modulation that respects the constraint for a given window size and to change the worst case modulation rate proportionally with the window size.

As the next step we need to define the worst case signal that is the signal that will be used to derive the descriptor limits of the sinusoidal class. From the wide range of possible modulation laws we have chosen the sinusoidal amplitude and frequency modulation in white Gaussian background noise as our worst case reference signal. The choice is motivated by the fact that a wide range of FM and AM conditions can be covered. If the window size is small compared to the vibrato rate for example, it is easy to see that the vibrato signal approximately creates linear FM and AM. Recent investigations have shown [11,12] that for real world vibrato signals the AM and FM will generally not be phase synchronous. Accordingly, the worst case signal model exhibits arbitrary phase relations between the amplitude and frequency modulation. A special feature of real world AM is the fact that the dominant AM rate may either be the same as the FM rate, or twice as high. As the letter case is more critical, we chose it for our worst case signal scenario.

In a view of the aforementioned discussion, the following mathematical expression for the sinusoidal model is proposed:

$$x(n) = \cos\left[2\pi F_0 n + A_{FM}\sin\left(2\pi F_{FM} n + \alpha\right)\right]\times \left[1 + A_{AM}\cos\left(2\pi F_{AM} n + \beta\right)\right] + r(n), \quad (11)$$

where $r(n)$ is additive Gaussian noise. The parameters are selected as follows. According to the previous discussion we set $F_{AM} = 2F_{FM}$. The frequency vibrato rate F_{FM} has to be selected such that the spectrum always contains a significant mainlobe, which is ensured by $F_{FM} = 1/(4.2M)$. Accordingly, the window covers less than the fourth part of the FM vibrato period. The values for the amplitude and frequency modulation depth have been chosen as $A_{AM} = 0.5$ and $A_{FM} = 10$.

These values ensure a dominant peak mainlobe for arbitrary phase angles (α and β) . The window length$M$, the sinusoidal frequency F_o, and the sample-rate R do not have any impact on the results. The size of the DFT N is chosen in such a way to assure that the Picket-Fence effect has minimal impact on a peak representation in the discrete spectrum. For completeness we note the values that we used for the following investigation into the descriptor distributions (M = 40ms, N = 4096, F_o = 880Hz, R = 44 kS/s).

It is clear that the present worst case signal does not cover all modulations that may be encountered in a real world setting, even if we respect the fact that a dominant mainlobe is required to detect a modulated sinusoid. The explicit inclusion of time varying sinusoids into the model will nevertheless lead to a classifier that has significant advantages in real world situations with time varying sinusoids.

Because the part of the sinusoidal peak that can be observed changes with the variance σ_r^2 of the background noise level $r(n)$ the peak descriptors will not only change with the modulation, but also with the SNR. For multicomponent signals the global SNR does not provide meaningful insight, and therefore, we will use the *Peak Signal-to-Noise Ratio* (SNR_p) as our noise level parameter. The SNR_p indicates the sinusoidal peak power level in dB over the noise floor (see Figure 1) and it presents a convenient parameter to control the limits of the sinusoidal class.

To experimentally create the descriptor distributions we proceed as follows. For the noise class distributions we calculate the descriptors for all spectral peaks in the DFT of white Gaussian noise processes using an analysis window of size M. For the sinusoidal class we create a grid of phase values covering all combinations α and β over the range $-\pi$ to π and we set $\sigma_r^2 = 0$. Then we calculate the descriptor values for the largest peak in each frame. This gives us the distributions for an infinite SNR_p. The sidelobe distributions are calculated from all but the strongest spectral peak in the spectrum of the worst case sinusoid. The resulting descriptor distributions are normalized by the maximum value and shown in Figure 2 for the Hanning window.

As we can see from Figure 2 the NBD distributions for modulated noise free sinusoidal peaks and for noise peaks do not overlap at all, making them a very good candidate for sinusoidal and noise separation. The sine and noise distributions for the NDD significantly overlap, but the sinusoidal distribution covers only a small range of descriptor values. This fact will be used to refine the sine/noise separation done by the NBD for signals of finite SNR_p as will be explained in the next section. Finally, the sidelobe structures can efficiently be distinguished by means of the FCD. Note that in Figure 2 the maximum of the sidelobe distribution is to be interpreted as a cumulus of all the sidelobe FCD values distributed out of the current axis range.

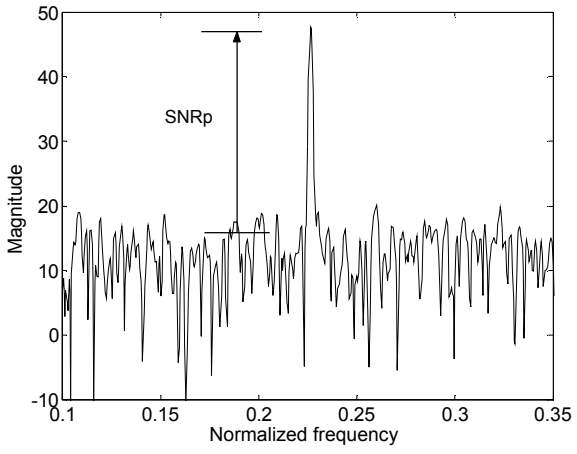


Figure 1: Illustration for the parameter SNR_p (peak signal-to-noise ratio)

4. CLASSIFICATION STRATEGY

The peak classification algorithm, based on the proposed peak descriptors, is established through a two-level decision tree as follows: in the first level the sidelobe and non-sidelobe classification is performed. Then in the second level the peaks previously declared non-sidelobes are classified as sinusoids and noise. The thresholds for both levels of classification are obtained by means of analyzing the distributions shown in Figure 2. For infinite SNR_p the classification could be obtained by simply using FCD and NBD thresholds to perfectly seperate all three peak classes. Note that only in this particular case the NBD attains almost perfect sine/non-sine classification, therefore the contribution of the NDD is negligible.

For a finite SNR_p the sinusoidal distributions experiment a spread proportional to the noise level in the worst case signal. In particular, the NBD sinusoidal distribution extends towards right while the NDD sinusoidal distribution spreads in both directions. The sinusoidal NBD distribution overlaps partially with the noise NBD distribution, which means that the NBD can no longer separate perfectly the peak classes. In order to reduce this ambiguity, we make use of the NDD. As mentioned before, the sinusoidal NDD distribution covers only a small range of descriptor values. Hence, by considering only the peaks within the limits of the sinusoidal NDD distribution as sinusoids, we can eliminate some of the noise peaks previously classified as sinusoids and thus refine the initial sine/noise classification. The classification scheme that is used for finite SNR_p is shown in Table 1. It is important to understand that a decreasing SNRp will modify the limits of the sinusoidal distribution in a similar manner as an increase in the modulation parameters would do. Therefore, the minimum SNRp can be used to control the decision thresholds in a rather intuitive manner.

In order to keep track of the limit values of the sinusoidal distributions we would need to regenerate all the sinusoidal distributions every time the minimum SNR_p that is selected by the user is changed. As shown below, however, the experimental evaluation of the distribution limits can be avoided, due to a simple approximate formula that expresses the relationship between the parameter SNR_p and the margins of the sinusoidal peak distributions in the descriptor domain. These can be used to adapt the classifier to the selected SNR_p. The thresholds to be adapted are the right margin of the NBD sinusoidal distribution and both margins of the NDD sinusoidal distribution. As for the FCD, the threshold may be kept fixed thanks to the good sidelobe separation from the rest of the peak classes.

sidelobe / non-sidelobe	$FCD \geq N/M$
sine / noise	$NBD \leq 0.13$ & $0.13 \leq NDD \leq 0.16$

Table 1: Peak classification thresholds for infinite SNR_p, the window is Hanning.

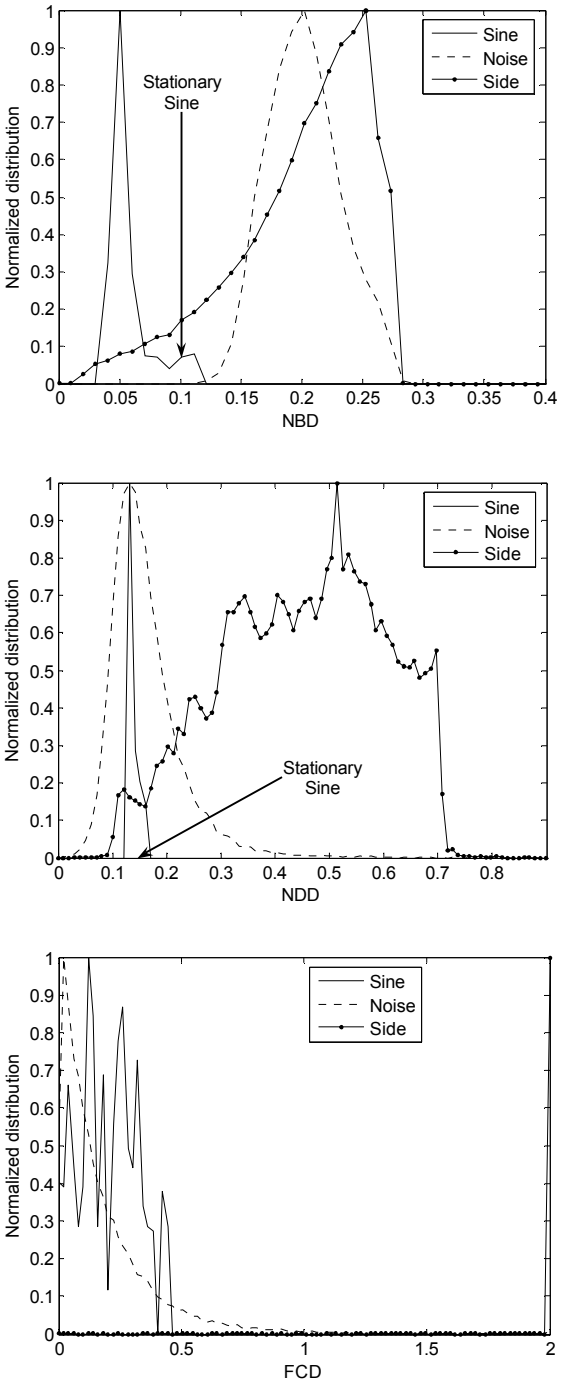


Figure 2: Normalized distributions for three peak classes in the descriptor domain; $\sigma_r^2 = 0$ and the window is Hanning.

5. MODELLING SNR_P DEPENDENCY

The relation between the classification threshold and the SNR_p is rather complex and to be able to achieve a model of these relations the problem requires a number of simplifications. The idea we pro-

pose is to first experimentally determine the signal pattern that is related to the descriptor limits for infinite SNR_p. Then we develop a simplified model of the effect of the additive noise to be able to achieve a mathematical formulation of the threshold dependency on the SNR_p. The relation does not take into account that the signal pattern at the descriptor limits may depend on the SNRp.

Window	α_{max}	β_{max}
Hanning	0.75π	0.50π
Blackman	0.75π	0.55π
Hamming	0.70π	0.45π

Table 2: The phase values of the sinusoidal model corresponding to NBD_{max} *for various analysis windows*

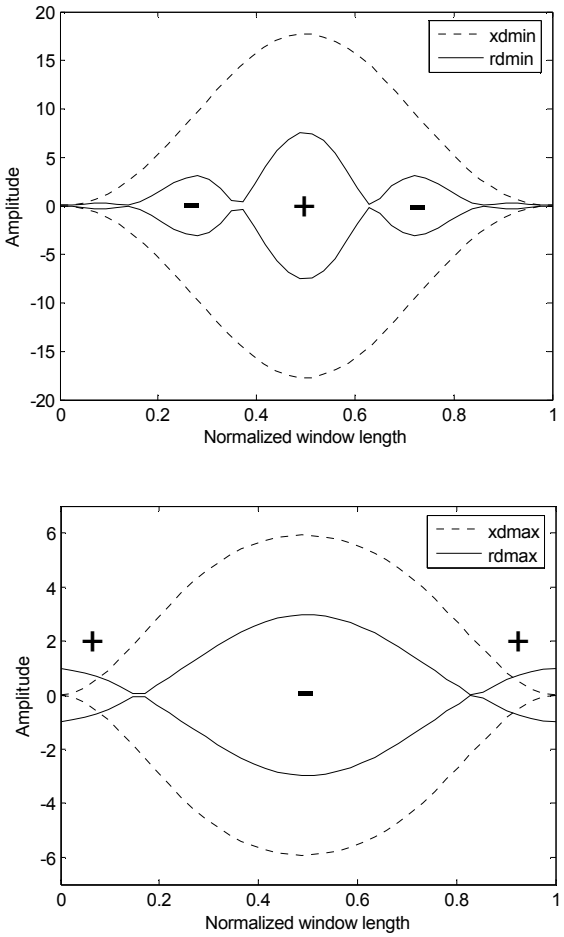


Figure 3: Envelopes of the signal patterns and noise patterns corresponding to the NDD thresholds for SNR_p = 10dB; the sign symbols mark the carrier phase relationship between the waveform; the analysis window is Hanning.

5.1. NBD threshold (NBD_{max})

Let us recall that the NBD is the ratio of the peak bandwidth and peak width. As described above we first need to determine the sinusoidal signal that will give rise to the maximum value of the descriptor $NBD_{max} = BW/L$. This can be done by means of a straightforward search over the two-dimensional grid of phase values α and β for a given analysis window (see Table 2 for some prominent analysis windows).

The presence of noise will affect both *BW* and *L*. It is clear that *L* will decrease because the peak local minima get closer to the peak maximum in terms of magnitude. In a simple approximation we may assume that *BW* will keep almost constant because the peak shape around the maximum is only slightly affected by additive noise. Accordingly, we may assume that the NBD_{max} is a function of *L* solely, which in turn depends on the SNR_p. Practically, for the given α_{max} and β_{max} we calculate the spectrum of the sinusoidal signal only once and store it in memory. Then, the NBD threshold can easily be calculated by taking into account only the DFT bins of the mainlobe that lie above the noise floor given by SNR_p. The validity of this simple approximation will be checked in the next section by comparing its values to those obtained by measuring NBD_{max} for different SNR_p and different analysis windows.

5.2. NDD threshold (NDD_{min} and NDD_{max})

The sinusoidal model in (11) is herein simplified in order to investigate into the NDD thresholds. More specifically, the FM can be disregarded because it does not modify the NDD of a sinusoid. Hence,

$$x(n) = \cos(2\pi F_0 n)\times \left[1 + A_{AM}\cos(2\pi F_{AM} n + \beta)\right] + r(n) \quad . \tag{12}$$

The phase β that gives rise to the minimum and maximum values of the NDD descriptor for the signal in (12) and after applying the analysis window can be calculated numerically. The solution shows that the maximum value is obtained when the minimum of the AM envelope is located in the signal center. The minimum of the NDD is obtained for a phase β that places the AM envelope maximum close to the window center. Due to the interactions between the analysis window and the envelope the AM envelope is not exactly aligned with the window center. To simplify the discussion and due to the fact that all values of beta in the range $-\pi \le \beta \le 0$ result in a variation of the NDD of less than 1% we will use the signal pattern with AM envelope maximum in the window center for the following discussion. Accordingly the (approximate) signal patterns for the shortest and longest signal in terms of the NDD are:

$$x_{d\min} = x(n;\beta = -0.5\pi)w(n),$$
$$x_{d\max} = x(n;\beta = 0.5\pi)w(n), \tag{13}$$

where $w(n)$ is the analysis window. The envelopes of the signal patterns $x_{d\min}$ and $x_{d\max}$ for the Hanning window are displayed in Figure 3. For finite SNR_p the patterns in (13) are superposed to a narrow-band Gaussian noise. Due to the small bandwidth of the signal peak the effective noise bandwidth is rather small. For each

SNR_p there exist two noise signal patterns, $r_{d\min}$ and $r_{d\max}$, that will maximally increase respectively decrease the NDD_{max} and NDD_{min} values. We will use a very simple signal model consisting of an amplitude modulated carrier as basis for our noise model. The noise model is band limited (reflecting the bandwidth of the spectral peak) but not necessarily time limited. Due to the small bandwidth the noise pattern may extend out of the signal window. Because for the simple model we are aiming at we don't want to take into account the length of the DFT we will limit the noise signal to the time segment of the analysis window.

In order to reduce NDD_{min} $r_{d\min}$ should narrow the width of the central maximum of $x_{d\min}$. To achieve this $r_{d\min}$ must be in-phase with $x_{d\min}$ around the window's centre and in counter-phase otherwise. Because a strong amplitude at the window boundaries would always enlarge the NDD we additionally assume that the noise pattern $r_{d\min}$ has the analysis window applied.

On the contrary, $r_{d\max}$ must be in counter-phase with $x_{d\max}$ around the window's centre and in-phase close to the window edges. The resulting waveform would have the energy more uniformly distributed along the analysis window and thus larger NDD_{max}. $r_{d\max}$ must not be tapered in order to contribute significantly to the energy concentration in $x_{d\max}$ around the window edges.

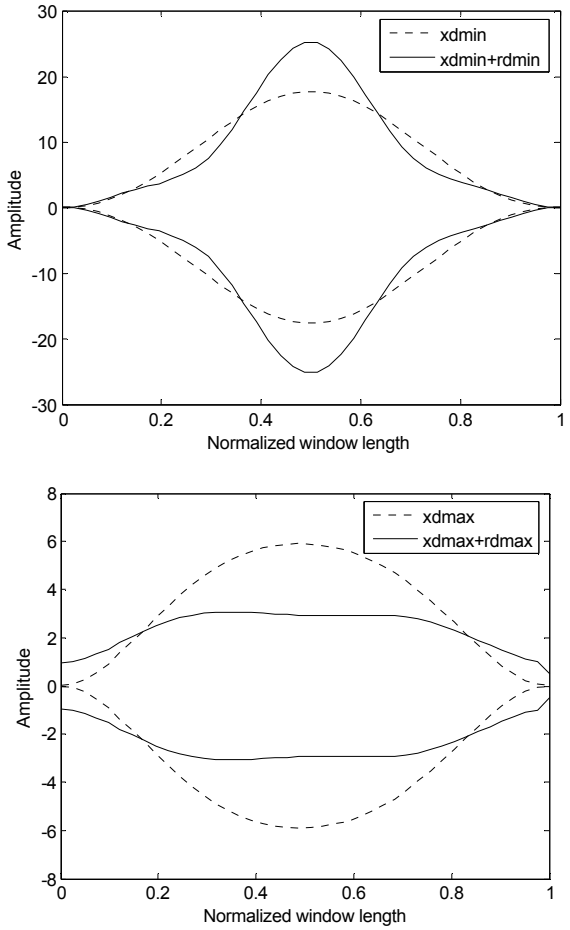


Figure 4: The resulting envelopes after the superposition of the signal patterns to the corresponding noise patterns for SNR_p = 10dB; the analysis window is Hanning.

According to the above discussion we have used the following model for the narrow-band Gaussian noise patterns:

$$r_{d\,\min}(n) = A\cos(2\pi F_o n)[1 + m_{\min}\cos(4\pi n/M)]w(n),$$

$$r_{d\,\max}(n) = -A\cos(2\pi F_o n)[1 - m_{\max}\cos(2\pi n/M)]. \quad (14)$$

The noise patterns are therefore sine-modulated waveforms. The modulation frequencies are different because the bandwidth of the peaks related to NDD_{min} and NDD_{max} are different. They have been selected such that they obey a simple relation to the window size. Note that the exact frequency values are not critical for the model and that the frequencies do not depend on the SNR_p.

The modulation indices $m_{\min}$ and $m_{\max}$ have to be greater than one in order to ensure the phase change of π in the crossover between contiguous modulation lobes. Both amplitude A and modulation indices are function of the SNR_p. A determines the total energy of each pattern while $m_{\min}$ and $m_{\max}$ control the distribution of that energy along the analysis window. The amplitude is simply a scaling factor that ensures the most of the spectral energy of the noise patterns lays SNR_p decibels under the mainlobe of the worst case signal. The values for the modulation indices are more difficult to estimate as they change in a non-linear fashion with the SNR_p. To obtain a mathematical model we have used the signal (13) and a wide range of SNR_p settings and have experimentally determined the maximum and minimum NDD as well as the values for $m_{\min}$ and $m_{\max}$ that would best match the experimental data. Finally, we derived a second order polynomial representation of the modulation indices by means of adapting a second order polynomial to the set of modulation indices. For various types of analysis windows the resulting functions are:

$$m_{\min} = \sum_i a_i SNR_p^i \;, \qquad m_{\max} = \sum_i b_i SNR_p^i \;,$$

while the corresponding coefficients are given in Table 3. For the Hanning window, the envelopes of the corresponding noise patterns for SNR_p = 10dB are shown on Figure 3 while the envelopes of the resulting waveforms after the superposition are shown on Figure 4. We can observe that the energy distributions of the signal patterns have indeed been modified coherently to the aforementioned explanation. In practical applications, the signal patterns are calculated only once while the noise patterns are recalculated each time the SNR_p or type of analysis window is changed such that the new thresholds can be obtained. We will show in the following section the behavior of this model with respect to the measured NDD_{min} and NDD_{max} for different SNR_p and various analysis window types.

6. EXPERIMENTAL RESULTS

In this section we aim to check the validity of the proposed adaptive threshold selection algorithm. For different types of analysis windows and for a wide range of SNR_p values, the decision thresholds NBD_{max}, NDD_{min} and NDD_{max} were generated from the corresponding models (Section 5) and compared to their respective measured values. The measured values are obtained from the Gaussian noise added to the sinusoidal model in the proportion established by the SNR_p. The approximation errors are calculated as a difference between the measured and modeled values and are shown on Figure 5. Generally,

the approximation errors are larger for smaller SNR_p. In case of the NBD_{max} and the NDD_{min} thresholds the experimentally obtained errors show a systematic trend. This could be used to refine the model. For the NDD thresholds the error is generally overestimating the change of the boundaries that goes with the SNRp. For the NBD threshold the threshold change is underestimated. The overall approximation error is obtained by evaluating the correlation coefficient R between the measured and approximated curve for each threshold and various analysis windows. From Table 4 we can see that in almost all situations the correlation coefficient is above 0.95 which can be considered a very good approximation. Also, note that the largest approximation errors are committed in the NBD_{max} thresholding domain for the Hanning window. On the contrary, the Blackman window thresholding adapts well to the corresponding curve of measured threshold values.

Window	a_i (r_{dmin})	b_i (r_{dmax})
Hanning	0.0174 -0.5770 10.6280	-0.0006 0.1211 0.8279
Blackman	0.0081 -0.3903 9.0630	-0.0022 0.1472 0.7083
Hamming	0.0003 -0.2816 2.4716	-0.0037 0.1615 0.7230

Table 3: The coefficient values for modeling the m_{min} and m_{max} dependency on SNRp.

Window	Hanning	Blackman	Hamming
$R(NDD_{min})$	0.9567	0.9685	0.9604
$R(NDD_{max})$	0.9799	0.9792	0.9840
$R(NBD_{max})$	0.9139	0.9885	0.9585

Table 4: The correlation coefficient calculated between the measured and approximation threshold curves for various analysis windows

7. CONCLUSIONS

In this paper we have presented a new adaptive threshold selection algorithm that can be used for classification of spectral peaks. By means of the set of peak descriptors from previous work and a herein proposed compact sinusoidal model related to the analysis window, the limit values for the distributions of sinusoidal peaks in the descriptor domain can be explicitly obtained. Next, the variations of those limit values, due to the presence of noise in the sinusoidal model, are characterized in a deterministic fashion through only one parameter we refer to as the peak signal/noise ratio. By means of this user-defined parameter the descriptor limits of the classification algorithm can be controlled intuitively using as control parameter the peak signal to noise ratio.

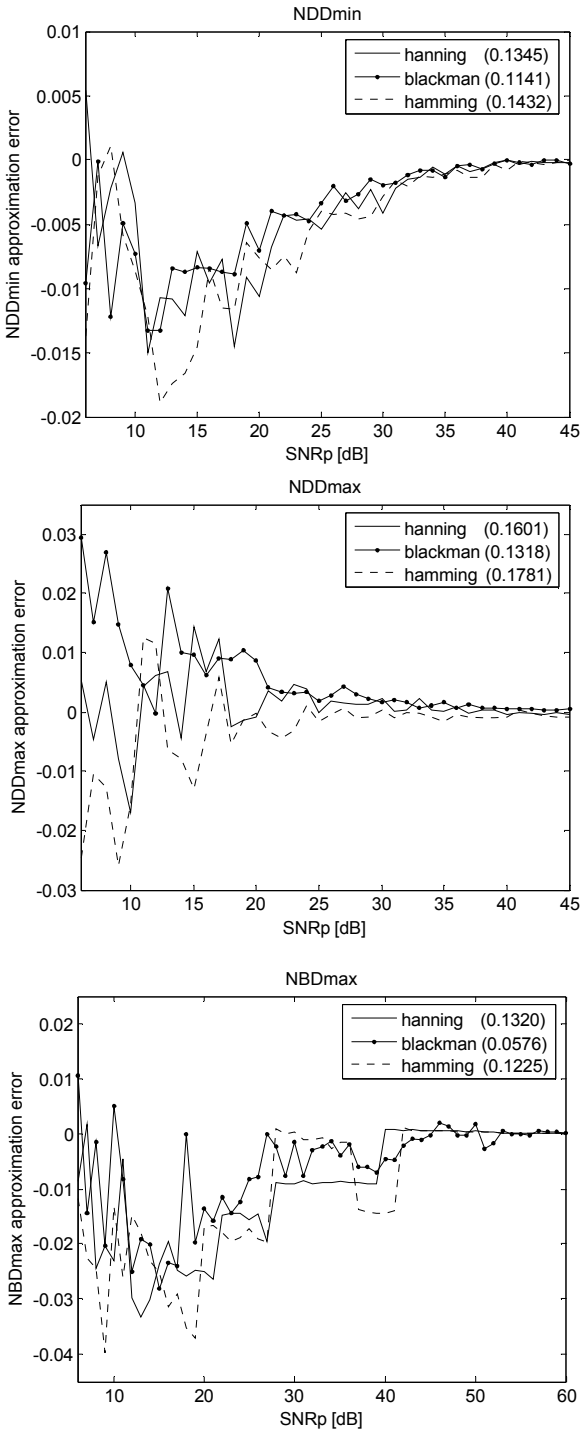


Figure 5: Approximation errors calculated as a difference between the measured and modeled values for each SNR_p and various analysis windows. The values in the legend correspond to infinite SNR_p.

The approximation accuracy given through the correlation coefficient is shown to be large for different types of analysis window. At the present state the new threshold selection method provides a control precision that can be considered sufficient for interactive control of a classification algorithm. Further investigation will be concerned with the improving the threshold models in order to reduce the approximation errors such that the precision of the control can be improved.

8. ACKNOWLEDGEMENTS

The first author of the paper would like to gratefully acknowledge the financial support of the Universidad Publica de Navarra, Spain.

9. REFERENCES

[1] A. Röbel, "A new approach to transient processing in the phase vocoder" in Proc. of the 6th Int. Conf. on Digital Audio Effects (DAFx03), 2003, pp. 344–349.

[2] M.Zivanovic, A. Röbel, X. Rodet, "A new approach to spectral peak classification", in Proc. of the 12th EUSIPCO, Vienna, Austria, September 2004, pp.1277-1280

[3] X. Rodet, "Musical sound signal analysis/synthesis: Sinusoidal + residual and elementary waveform models," in Proc IEEE Time-Frequency and Time-Scale Workshop 97, (TFTS'97), 1997

[4] D. J. Thompson, "Spectrum estimation and harmonic analysis", IEEE Proc. Vol.70, No.9, September 1982

[5] C. Yeh, A. Röbel, X. Rodet, 'Multiple Fundamental Frequency Estimation Of Polyphonic Music Signals", Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP 2005), pp. 225-228, Vol III, 2005.

[6] C. Yeh, A. Röbel, "Adaptive noise level estimation", Proc. of the 9th Int. Conf. on Digital Audio Effects (DAFx'06), Montreal, pp. 145-148, 2006.

[7] P. Depalle, G. Garcia, X. Rodet, "Tracking of partials for additive sound synthesis using hidden Markov models," in Proc. Int. Conf. on Acoustics, Speech and Signal Processing, vol. I, pp. 242–245, 1993

[8] L.Cohen, "Time-frequency analysis", Prentice Hall, 1995

[9] F. Auger and P. Flandrin, "Improving the readability of time-frequency and time-scale representations by the reassignment method," IEEE Trans. on Signal Processing, vol. 43, no. 5, pp. 1068–1089, 1995.

[10] A. Röbel, "Adaptive additive modeling with continuous parameter trajectories", IEEE Transactions on Speech and Audio Processing, Vol. 14, No. 4, pp.1440-1453, 2006.

[11] V. Verfaille, C. Guastavino, P. Depalle, "Perceptual evaluation of vibrato models", Proc. of the Conf. on Interdisciplinary Musicology (CIMOS), 2006

[12] I. Arroabarren, X. Rodet, A. Carlosena, "On the measurement of the instantaneous frequency and amplitude of partials in vocal vibrato", ", IEEE Transactions on Speech and Audio Processing, Vol. 14, NO. 4, pp.1413-1421, July 2006

REAL-TIME AUDIO PROCESSING VIA SEGMENTED WAVELET TRANSFORM

Pavel Rajmic and Jan Vlach

Dept. of Telecommunications
FEEC, Brno University of Technology
Czech Republic
rajmic@feec.vutbr.cz

ABSTRACT

In audio applications it is often necessary to process the signal in "real time". The method of segmented wavelet transform (SegWT) makes it possible to compute the discrete-time wavelet transform of a signal segment-by-segment, not using the classical "windowing". This means that the method could be utilized for wavelet-type processing of an audio signal in real time, or alternatively in case we just need to process a long signal, but there is insufficient computational memory capacity for it (e.g. in the DSPs). In the paper, the principle of the segmented forward wavelet transform is explained and the algorithm is described in detail.

1. INTRODUCTION

There are a number of theoretical papers and practical applications of the wavelet transform. However, all of them approach the problem from such a point of view as if we knew the whole of the signal (no matter how long it is). Due to this assumption, we cannot perform the wavelet-type signal processing in real time in this sense. Of course there are real-time applications of the wavelet type, but, all of them utilize the principle of overlapping segments of the "windowed" signal (e.g. [1]). In the reconstruction part of their algorithms they certainly introduce errors into the processing, because the segments are assembled using weighted averages.

Processing a signal in "real time" actually means processing it with minimum delay. A signal, which is not known in advance, usually comes to the input of a system piecewise, by mutually independent segments that have to be processed.

The new method, the so-called segmented wavelet transform (SegWT – we introduce abbreviation SegWT (Segmented Wavelet Transform), because SWT is already reserved for stationary wavelet transform), enables this type of processing. It has a great potential application also in cases when it is necessary to process a long signal off-line and no sufficient memory capacity is available. It is then possible to use this method for equivalent segmentwise processing of the signal and thus save the storage space. In this sense SegWT corresponds to the overlap-add algorithm in Fourier-type linear filtering.

Another possible application of the SegWT algorithm is the instantaneous visualization of signal using an imaging technique referred to as "scalogram", see Fig. 1. The decomposition depth is $d = 5$ in this Figure. The bigger is the absolute value of the single coefficient, the whiter is the respective cell in the graph. In fact, plotting scalogram is a technique very similar to plotting a spectrogram in real time. In wavelet transformation (represented by FIR filters) there is an advantage in that the signal need not be weighted with windows, which results in a distortion of the frequency information, as is the case with the spectrogram. Moreover, there is

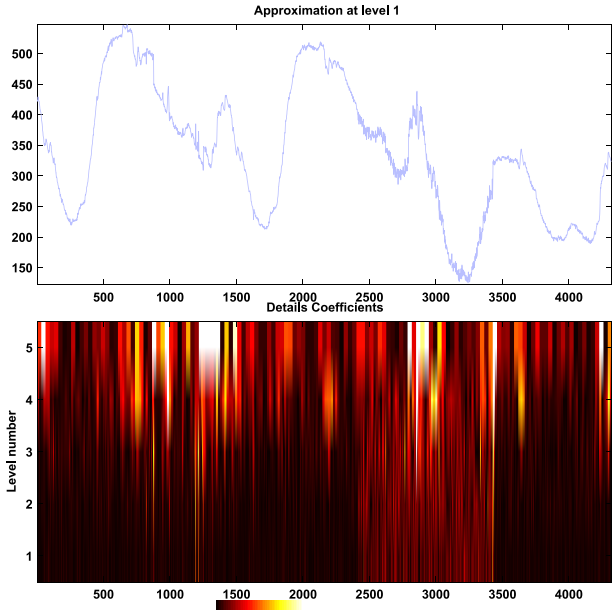


Figure 1: Signal (top) and its scalogram (bottom). Scalogram is a type of graph representing the frequency contents of a signal in time. It is constructed from the wavelet coefficients.

one more good thing about it: a scalogram created by means of the SegWT is quite independent of the chosen length of segment.

In the available literature, this way of performing the wavelet transform is practically neglected, and this was the reason why our effort was devoted to developing modified algorithm. In fact, a modified method of *forward* wavelet transform is presented in this paper.

2. THE CLASSICAL DTWT ALGORITHM

Algorithm 2.1: (decomposition pyramidal algorithm DTWT) Let $\mathbf{x}$ be a discrete input signal of length s, the two wavelet decomposition filters of length m are defined, highpass $\mathbf{g}$ and lowpass $\mathbf{h}$, d is a positive interger determining the decomposition depth. Also, the type of boundary treatment [2, ch. 8] must be known.

1. We denote the input signal $\mathbf{x}$ as $\mathbf{a}^{(0)}$ and set $j = 0$.
2. One decomposition step:

(a) *Extending the input vector.* We extend $\mathbf{a}^{(j)}$ from both the left and the right side by $(m-1)$ samples, according to the type of boundary treatment.

(b) *Filtering.* We filter the extended signal with filter $\mathbf{g}$, which can be expressed by their convolution.

(c) *Cropping.* We take from the result just its central part, so that the remaining "tails" on both the left and the right sides have the same length $m-1$ samples.

(d) *Downsampling (decimation).* We downsample the resultant vector.

We denote the resulting vector $\mathbf{d}^{(j+1)}$ and store it.We repeat items (b)–(d), now with filter $\mathbf{h}$, denoting and storing the result as $\mathbf{a}^{(j+1)}$.

3. We increase j by one. If it now holds $j < d$, we return to item 2., in the other case the algorithm ends.

Remark. After algorithm 2.1 has been finished, we hold the wavelet coefficients stored in $d+1$ vectors $\mathbf{a}^{(d)}, \mathbf{d}^{(d)}, \mathbf{d}^{(d-1)}, \ldots, \mathbf{d}^{(1)}$.

3. THE METHOD OF SEGMENTED WAVELET TRANSFORM

3.1. Motivation and Aim of the Method

Regularly used discrete-time wavelet transform (see Section 2) is suitable for processing signals "off-line", i.e. known before processing, even if very long. The task for the segmented wavelet transform, SegWT, is naturally to allow signal processing by its segments, so that in this manner we get the same result (same wavelet coefficients) as in the ordinary DTWT case. In this problem, the following parameters play a crucial role.

m wavelet filter length, $m > 0$,
d transform depth, $d > 0$,
s length of segment, $s > 0$.

The derivation of the SegWT algorithm requires a very detailed knowledge of the DTWT algorithm. Thanks to this it is possible to deduce fairly sophisticated rules how to handle the signal segments. We have found that in dependence on m, d, s, it is necessary to extend every segment from the left by an exact number of samples from the preceding segment and from the right by another number of samples from the subsequent segment. However, every segment has to be extended by a different length from the left and the right, and these lengths can also differ from segment to segment! Also the first and the last segments have to be handled in a particular way.

3.2. Important Theorems Derived from the DTWT Algorithm

Before we introduce detailed description of the SegWT algorithm, several theorems must be presented. More of them and their proofs can be found in [3, ch. 8]. We assume that the input signal $\mathbf{x}$ is divided into $S \geq 1$ segments of equal length s. Single segments will be denoted ${}^1\mathbf{x}, {}^2\mathbf{x}, \ldots, {}^S\mathbf{x}$. The last one can be of a length lower than s. See Fig. 2.

By the formulation that *two sets of coefficients from the k-th decomposition level follow-up on each other* we mean a situation when two consecutive segments are properly extended see Figs. 2, 3, so that applying the DTWT, with step 2(a) omitted, of depth

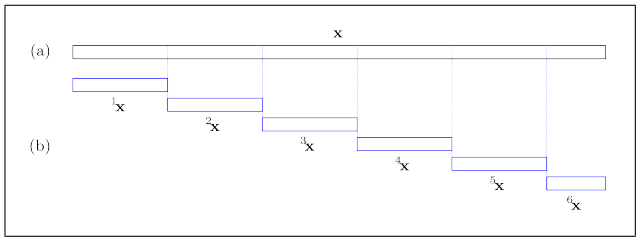


Figure 2: Scheme of signal segmentation. The input signal $\mathbf{x}$ (a) is divided into segments of equal length, the last one can be shorter than this (b); the n-th segment of $\mathbf{x}$ is denoted by ${}^n\mathbf{x}$.

k separately to both the segments and joining the resultant coefficients together lead to the same set of coefficients as computing it via the DTWT applied to the two segments joined first.

Theorem 3.1: *In case that the consecutive segments have*

$$r(k) = (2^k - 1)(m-1) \tag{1}$$

common input signal samples, the coefficients from the k-th decomposition level follow-up on each other.

Thus, for a decomposition depth equal to d it is necessary to have $r(d) = (2^d - 1)(m-1)$ common samples in the two consecutive extended segments.

The aim of the following part is to find the proper extension of every two consecutive signal segments. We will show that the length of such extension must comply with the strict rules.

The extension of a pair of consecutive segments, which is of total length $r(d)$, can be divided into the right extension of the first segment (of length R) and the left extension of the following segment (of length L), while $r(d) = R + L$. However, the lengths $L \geq 0, R \geq 0$ cannot be chosen arbitrarily. The lengths L, R are not uniquely determined in general. The formula for the choice of extension $L_{\max}$, which is unique and the most appropriate in case of real-time signal processing, is given in Theorem 3.2.

Theorem 3.2: *Let a segment be given whose length including its left extension is l. The maximal possible left extension of the next segment, $L_{\max}$, can be computed by the formula*

$$L_{\max} = l - 2^d \operatorname{ceil}\left(\frac{l - r(d)}{2^d}\right). \tag{2}$$

The minimal possible right extension of the given segment is then

$$R_{\min} = r(d) - L_{\max}. \tag{3}$$

For the purposes of the following text, it will be convenient to assign the number of the respective segment to the variables $L_{\max}, R_{\min}, l$, i.e. the left extension of the n-th segment will be of length $L_{\max}(n)$, the right extension will be of length $R_{\min}(n)$ and the length of the original n-th segment with the left extension joined will be denoted $l(n)$. Using this notation we can rewrite equation (3) as

$$R_{\min}(n) = r(d) - L_{\max}(n+1). \tag{4}$$

Let us now comment on the special situation of the first or the last segment. These naturally represent the "boundaries" of the signal. The discrete-time wavelet transform uses several modes how to treat the boundaries and we must preserve these modes also in our modified algorithm. Therefore we must treat the first

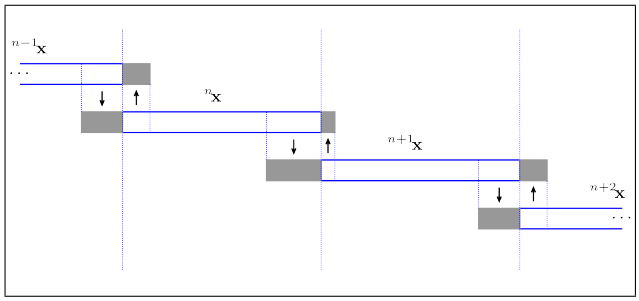


Figure 3: Illustration of extending of the segments.

and the last segment separately and a bit differently from the other segments. For details and proofs we again refer to [3]. The appropriate procedure is to extend the first segment from the left by $r(d)$ zero samples, i.e. $L_{\max}(1) = r(d)$, and to process it using Algorithm 3.7. Similarly the last segment has to be extended by $r(d)$ zeros from the right and processed using Algorithm 3.8.

Theorem 3.3: *The length of the right extension of the n-th segment, $n = 1, 2, \ldots, S-2$, must comply with*

$$R_{\min}(n) = 2^d \operatorname{ceil}\left(\frac{ns}{2^d}\right) - ns, \tag{5}$$

and the length of the left extension of the $(n+1)$-th segment is $L_{\max}(n+1) = r(d) - R_{\min}(n)$.

Remark. From (5) it is clear that $R_{\min}$ is periodic with respect to s with period 2^d, i.e. $R_{\min}(n + 2^d) = R_{\min}(n)$. This relation and also some more can be seen in Table 1.

Theorem 3.4: ***(on the total length of segment)***
After the extension the n-th segment (of original length s) will be of total length

$$\sum(n) = r(d) + 2^d \left[\operatorname{ceil}\left(\frac{ns}{2^d}\right) - \operatorname{ceil}\left(\frac{(n-1)s}{2^d}\right)\right]. \tag{6}$$

This expression can acquire only one of two values, either

$$r(d) + 2^d \operatorname{ceil}\left(\frac{s}{2^d}\right) \quad \textit{or} \quad r(d) + 2^d \operatorname{ceil}\left(\frac{s}{2^d}\right) - 2^d. \tag{7}$$

3.3. The Algorithm of Segmented Wavelet Transform

The algorithm SegWT works such that it reads (receives) single segments of the input signal, then it extends – overlaps them in a proper way, then it computes the wavelet coefficients in a modified way and, in the end, it easily joins the coefficients.

Algorithm 3.5: Let the wavelet filters $\mathbf{g}, \mathbf{h}$ of length m, the decomposition depth d, and the boundary treatment mode be given. The segments of length $s > 0$ of the input signal $\mathbf{x}$ are denoted ${}^1\mathbf{x}, {}^2\mathbf{x}, {}^3\mathbf{x}, \ldots$. The last segment can be shorter than s.

1. Set $N = 1$.
2. Read the first segment, ${}^1\mathbf{x}$, and label it "current". Extend it from the left by $r(d)$ zero samples.
3. ***If*** the first segment is at the same time the last one
   (a) It is the case of regular wavelet transform. Compute the DTWT of this single segment using Algorithm 2.1.
   (b) The Algorithm ends.
4. Read $(N+1)$-th segment and label it "next".
5. ***If*** this segment is the last one
   (a) Join the current and next segment together and label it "current". (The current segment becomes the last one now.)
   (b) Extend the current vector from the right by $r(d)$ zero samples.
   (c) Compute the DTWT of depth d from the extended current segment using Algorithm 3.8.

   Otherwise
   (d) Compute $L_{\max}$ for the next segment and $R_{\min}$ for the current segment (see Theorem 3.2).
   (e) Extend the current segment from the right by $R_{\min}$ samples taken from the next segment. Extend the next segment from the left by $L_{\max}$ samples taken from the current segment.
   (f) ***If*** the current segment is the first one, compute the DTWT of depth d from the extended current segment using Algorithm 3.7. ***Otherwise*** compute the DTWT of depth d from the extended current segment using Algorithm 3.6.
6. Modify the vectors containing the wavelet coefficients by trimming off a certain number of redundant coefficients from the left side, specifically:at the k-th level, $k = 1, 2, \ldots, d-1$, trim off $r(d-k)$ coefficients from the left.
7. ***If*** the current segment is the last one, then in the same manner as in the last item trim the redundant coefficients, this time from the right.
8. Store the result as ${}^N\mathbf{a}^{(d)}, {}^N\mathbf{d}^{(d)}, {}^N\mathbf{d}^{(d-1)}, \ldots, {}^N\mathbf{d}^{(1)}$.
9. ***If*** the current segment is not the last one
   (a) Label the next segment "current".
   (b) Increase N by 1 and go to item 4.

Remark. If the input signal has been divided into $S > 1$ segments, then $(S-1)(d+1)$ vectors of wavelet coefficients

$$\{\, {}^i\mathbf{a}^{(d)},\ {}^i\mathbf{d}^{(d)},\ {}^i\mathbf{d}^{(d-1)}, \ldots, {}^i\mathbf{d}^{(1)} \}_{i=1}^{S-1}.$$

are the output of the Algorithm. If we join these vectors together in a simple way, we obtain a set of $d+1$ vectors, which are identical with the wavelet coefficients of signal $\mathbf{x}$.

Next we present the "subalgorithms" of the SegWT method. The second and third algorithms serve to process the first and the last segment.

Algorithm 3.6: This algorithm is identical with Algorithm 2.1 with the exception that we omit step 2(a), i.e. we do not extend the vector.

Algorithm 3.7: This algorithm is identical with Algorithm 2.1 with the exception that we replace step 2(a) by the step:

> *Modify the coefficients of vector $\mathbf{a}^{(j)}$ on positions $r(d-j) - m + 2, \ldots, r(d-j)$, as it corresponds to the given boundary treatment mode.*

Algorithm 3.8: This algorithm is identical with Algorithm 2.1 with the exception that we replace step 2(a) by the step:

> *Modify the coefficients of vector $\mathbf{a}^{(j)}$ on positions $r(d-j) - m + 2, \ldots, r(d-j)$, as it corresponds to the given boundary treatment mode, however this time taken* from the right side *of $\mathbf{a}^{(j)}$.*

s	n	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12** ...
512	$L_{\max}(n)$	105	105	105	105	105	105	105	105	105	105	105	105 ...
	$R_{\min}(n)$	0	0	0	0	0	0	0	0	0	0	0	0 ...
	$\sum(n)$	617	617	617	617	617	617	617	617	617	617	617	617 ...
513	$L_{\max}(n)$	105	98	99	100	101	102	103	104	105	98	99	100 ...
	$R_{\min}(n)$	7	6	5	4	3	2	1	0	7	6	5	4 ...
	$\sum(n)$	625	617	617	617	617	617	617	617	625	617	617	617 ...
514	$L_{\max}(n)$	105	99	101	103	105	99	101	103	105	99	101	103 ...
	$R_{\min}(n)$	6	4	2	0	6	4	2	0	6	4	2	0 ...
	$\sum(n)$	625	617	617	617	625	617	617	617	625	617	617	617 ...
515	$L_{\max}(n)$	105	100	103	98	101	104	99	102	105	100	103	98 ...
	$R_{\min}(n)$	5	2	7	4	1	6	3	0	5	2	7	4 ...
	$\sum(n)$	625	617	625	617	617	625	617	617	625	617	625	617 ...
516	$L_{\max}(n)$	105	101	105	101	105	101	105	101	105	101	105	101 ...
	$R_{\min}(n)$	4	0	4	0	4	0	4	0	4	0	4	0 ...
	$\sum(n)$	625	617	625	617	625	617	625	617	625	617	625	617 ...
517	$L_{\max}(n)$	105	102	99	104	101	98	103	100	105	102	99	104 ...
	$R_{\min}(n)$	3	6	1	4	7	2	5	0	3	6	1	4 ...
	$\sum(n)$	625	625	617	625	625	617	625	617	625	625	617	625 ...
518	$L_{\max}(n)$	105	103	101	99	105	103	101	99	105	103	101	99 ...
	$R_{\min}(n)$	2	4	6	0	2	4	6	0	2	4	6	0 ...
	$\sum(n)$	625	625	625	617	625	625	625	617	625	625	625	617 ...
519	$L_{\max}(n)$	105	104	103	102	101	100	99	98	105	104	103	102 ...
	$R_{\min}(n)$	1	2	3	4	5	6	7	0	1	2	3	4 ...
	$\sum(n)$	625	625	625	625	625	625	625	617	625	625	625	625 ...
520	$L_{\max}(n)$	105	105	105	105	105	105	105	105	105	105	105	105 ...
	$R_{\min}(n)$	0	0	0	0	0	0	0	0	0	0	0	0 ...
	$\sum(n)$	625	625	625	625	625	625	625	625	625	625	625	625 ...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮ ⋱

Table 1: Example – lengths of extensions for different lengths of segments s. The depth of decomposition is $d = 3$ and the filter length is $m = 16$.

3.4. Corollaries and Limitations of the SegWT Algorithm

In [3] there can be found several practical corollaries for SegWT, e.g. that the segments cannot be shorter then 2^d.

From the description in the above sections it should be clear that the time lag of Algorithm 3.5 is one segment (i.e. s samples) plus the time needed for the computation of the coefficient from the current segment. In a special case when s is divisible by 2^d it holds even $R_{\min}(n) = 0$ for every $n \in \mathbb{N}$ (see Theorem 3.3), i.e. the lag is determined only by the computation time!

3.5. A Few Examples

- For $d = 4$ and $m = 12$, the minimum segment length is just 16 samples. When we set $s = 256$, $R_{\min}$ will always be zero and $L_{\max} = r(4) = 165$. The length of every extended segment will be $256 + 165 = 421$ samples.
- For $d = 5$ and $m = 8$, the minimum segment length is 32 samples. When we set $s = 256$, $R_{\min}$ will always be zero and $L_{\max} = r(5) = 217$. The length of every extended segment will be $256 + 217 = 473$ samples.
- For $d = 5$ and $m = 8$ we set $s = 300$, which is not divisible by 2^5. Thus $R_{\min}$ and $L_{\max}$ will alternate such that $0 \leq R_{\min} \leq 31$ and $186 \leq L_{\max} \leq 217$. The length of every extended segment will be $300 + r(5) = 473$ samples.

3.6. Implementation

The SegWT algorithm had been implemented in C++ and its functionality had been verified. We implemented a simple "band-stop filter" as a VST plug-in module utilizing the SegWT Algorithm and a reduced version of the inverse transform. The testing of the efficiency showed that the most demanding part of the algorithm is the computation of the convolution which must be done in each stage of the transform.

4. CONCLUSION

The paper contains a description of the algorithm which allows us to perform the wavelet transform in real time. The algorithm works on the basis of calculating the optimal extension (overlap) of signal segments, and subsequent performance of the modified transform.

In the future it would be convenient to improve the computational effectivity by reducing redundant computations at the borders of the segments, as it follows from the Algorithm 3.5. Also, it should not be very difficult to generalize the SegWT method to include biorthogonal wavelets and more general types of decimation [4, 5], because the parameters of SegWT can be chosen in a fairly general way.

Another important part of the future work is the derivation of an efficient counterpart to the introduced method – the segmented *inverse* transform. In fact, we made first experience, in which it turned out, above all, that the time lag in the consecutive forward-inverse processing will be, unfortunately, always nonzero.

5. ACKNOWLEDGEMENTS

The paper was prepared within the framework of No. 102/06/P407, No. 102/06/1233 and No. 102/07/1303 projects of the Grant Agency of the Czech Republic and No. 1ET301710509 project of the Czech Academy of Sciences.

6. REFERENCES

[1] D. Darlington, L. Daudet and M. Sandler, "Digital Audio Effects in the Wavelet Domain." In *Proc. of the 5th Int. Conf. on Digital Audio Effects (DAFX-02)*, Hamburg (2002)

[2] G. Strang. and T. Nguyen, *Wavelets and Filter Banks*. Wellesley Cambridge Press (1996)

[3] P. Rajmic, *Exploitation of the wavelet transform and mathematical statistics for separation signals and noise, (in Czech)*, PhD Thesis, Brno University of Technology, Brno (2004)

[4] P. Dutilleux, "An implementation of the "algorithme à trous" to compute the wavelet transform." In *Wavelets: Time-Frequency Methods and Phase Space*, Inverse Problems and Theoretical Imaging, editors J.-M. Combes, A. Grossman, P. Tchamitchian. pp. 298–304, Springer-Verlag, Berlin (1989)

[5] G.P. Nason and B.W. Silverman, The stationery wavelet transform and some statistical applications. In *Wavelets and Statistics*, volume 103 of *Lecture Notes in Statistics*, editors A. Antoniadis, G. Oppenheim, pp. 281–300, Springer-Verlag, New York (1995)

STATISTICAL MEASURES OF EARLY REFLECTIONS OF ROOM IMPULSE RESPONSES

Rebecca Stewart and Mark Sandler

Centre for Digital Music, Queen Mary, University of London
London, UK
{rebecca.stewart|mark.sandler}@elec.qmul.ac.uk

ABSTRACT

An impulse response of an enclosed reverberant space is composed of three basic components: the direct sound, early reflections and late reverberation. While the direct sound is a single event that can be easily identified, the division between the early reflections and late reverberation is less obvious as there is a gradual transition between the two.

This paper explores two statistical measures that can aid in determining a point in time where the early reflections have transitioned into late reverberation. These metrics exploit the similarities between late reverberation and Gaussian noise that are not commonly found in early reflections. Unlike other measures, these need no prior knowledge about the rooms such as geometry or volume.

1. INTRODUCTION

A room can be assumed to be a linear time-invariant system where the impulse response (IR) of the system can be found by recording a broadband signal within the room. Often the IR is convolved with non-reverberant audio to add artificial reverberation by simulating recording the audio in the room.

An IR of a space consists of direct sound, early reflections, and late reverberation. The early reflections are a set of discrete reflections whose density increases until individual reflections can no longer be discriminated and/or perceived. While the direct sound is a single event that can be easily identified, the early reflections and late reverberation of an IR are more difficult to label. For this paper, the transition time of an IR will be the earliest point in time when the density of the reflections has reached a perceptual threshold in which individual reflections can no longer be distinguished.

Since the seminal publications by Schroeder [1] and Moorer [2], digital artificial reverberators have contained a component intended to create discrete echoes in order to simulate early reflections and a component whose intent is to create a set of reflections as dense as possible. Later developments with feedback delay networks acknowledge that prior to high frequency attenuation, the reverberator should produce white noise [3] and Moorer in [2] first discussed using frequency-shaped Gaussian noise to simulate the energy in late reverberation. The transition from early reflections to late reverberation can then be modeled as a deterministic system that transitions into a stochastic one [4]. The statistics regarding early reflections are of greater concern here because the statistics of late reverberation has been covered in depth, particularly in [3]. Abel and Huang [5] have also recently explored similar statistics as are examined here, looking at measures of reverberation quality, particularly for judging artificial reverberation.

The late reverberation of an IR tends towards a normal distribution, unlike the energy from early reflections. A progression towards a more normal distribution occurs as time increases and the acoustic energy within the space becomes more mixed [4]. A measurement of distribution can then be used to determine whether a point in time is more or less deterministic. That is, whether it is within the early reflections or late reverberation.

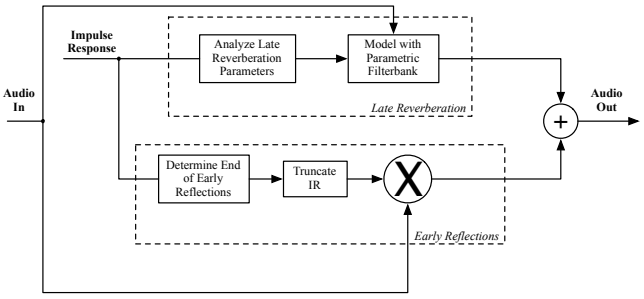


Figure 1: Basic design of a hybrid reverberator.

The ability to determine the mixing time of a space is especially relevant to hybrid reverberation. Hybrid reverberation uses both convolution and recursive filterbank techniques, as can be seen in Fig. 1. An IR is truncated to ideally contain the early reflections. The truncated IR is convolved with the dry audio and then the late reverberation is simulated with a filterbank. A precise measure of when a room is first mixed is important so that all perceptually relevant information is preserved in the early reflections of the truncated IR while the size of the truncated IR is as small as possible to reduce the length of the convolution.

1.1. Definitions of Early Reflections

Early reflections are loosely defined as a set of echoes that have not reached a perceptual threshold, and that can be described by the mathematical relationships that define the dispersion of echoes in a space such as

$$\frac{dN_r}{dt} = 4\pi \frac{c^3 t^2}{V} \tag{1}$$

where N_r is the number of reflections, t is the time from the direct sound, c is the speed of sound, and V is the volume of the room [6].

In [4], Blesser describes the mixing time as "how long it takes for there to be no memory of the initial state of the system. There is statistically equal energy in all regions of the space after the mixing time." He estimates this to be approximately three times the mean free path and directly a property of the geometry of the room. Here the mixing time is accepted to be after the transition from early reflections to late reverberation is complete. The upper

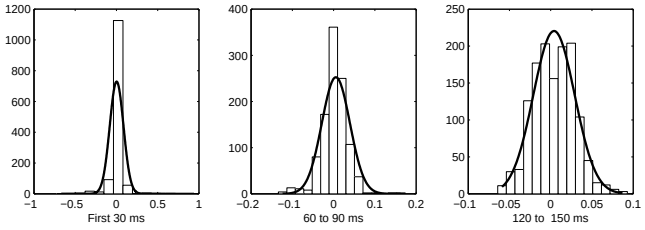


Figure 2: Progression in time of histograms and fit normal functions of 30 ms windows of an IR. Note the differing vertical and horizontal scales. The absolute values of the vertical scale are not important, but rather the relative values between the histogram bars. The range of the horizontal scale decreases in subsequent windows because the signal is decreasing in amplitude. It is important to note the distribution of the samples, not the absolute values.

limit of mixing time has been discussed in [7] and further in [3] to be

$$t_{mixing} = \sqrt{V} \tag{2}$$

where V is the volume in m^3.

Standardized measurements of room acoustics divide the IR of a room into an early and late portion in order to calculate early lateral energy, clarity and definition. When measuring for musical material, the early portion is defined as the first 80 ms [8]. In most literature, it is accepted that the early reflections are contained within the first 80 ms [9].

The transition from early reflections from late reverberation is either defined by a point in time regardless of the room properties, usually 80 ms, or is calculated based on physical properties of a room, most commonly volume. Using a single point in time regardless of the space is inaccurate, but access to the dimensions of a space may be impractical or impossible. A blind method that can determine the transition point between early reflections and late reverberation without knowledge of the measurements of the space is needed. Two possible methods are described here.

2. MEASURES OF DISPERSION

The transition from early reflections to late reverberation can be observed in several domains. This paper only addresses the transition with regard to a Gaussian distribution in the time domain, but other factors can be considered. As discussed in [10] and [11], the frequency distribution tends towards a Rayleigh distribution. As only monophonic impulse responses are studied here, the spatial properties of the room are not being considered.

2.1. Standard Deviation

The standard deviation of a group of samples is a measure of the spread of the samples and is defined as

$$\sigma = \sqrt{\mathbf{E}(x^2) - (\mathbf{E}(x))^2} \tag{3}$$

where $\mathbf{E}(x)$ is the expected value of x.

In a normal distribution, approximately one third of the samples lie outside one standard deviation of the mean and approximately two thirds of the samples are within one standard deviation of the mean. Early reflections, however, have more samples within one standard deviation and fewer outside (see Fig. 2). The progression from early reflections to late reverberation can then be observed through the ratio of samples outside one standard deviation versus inside.

As an IR progresses in time the ratio of samples gradually approaches approximately one third. Fig. 3 shows the ratio outside one standard deviation versus inside for a 30 ms window. The ratio is divided by $efrc(1/\sqrt{2})$, the expected value of samples outside one standard deviation, to normalize for Gaussian distribution. This is similar to what is done in [5].

Examples for three different spaces can be seen in Fig. 3. Fig. 3(a) is a measurement from a smaller space, a 350 seat concert hall, than those in Fig. 3(b) and 3(c) which are both large, reverberant churches. The measuring of the IRs from the churches is described in [12]; the concert hall was measured by the authors using the same technique and equipment.

The normalized ratio outside versus inside one standard deviation approaches one as time progesses, but it makes for a poor measure. It is difficult to identify the point in time when it can first be assumed that the room is mixed. While the curve approaches a threshold, that threshold is not constant amongst IRs from different spaces and is difficult to extrapolate from the curve. However, it is clear that the IR does gradually transition from less diffuse early reflections to more diffuse late reverberation. Using the ratio of samples outside to inside one standard deviation does not allow an easy selection of a transition point.

3. HIGHER ORDER STATISTICS

Moments describe deterministic signals as they "are numerical measures of the degree of similarity between a signal and a product of delayed or advanced versions of itself" [13]. An nth order cumulant is a function of its joint moment of orders up to n [14]. Higher order cumulants contain amplitude and phase information unlike second order statistics (correlation) which are phase-blind. The second order cumulant is the variance, the third order is skewness and fourth order is kurtosis [13].

While moments describe deterministic signals, cumulants are measures for stochastic signals. If a set of random variables are jointly Gaussian, then all information about their distribution is in the moments of an order less than or equal to two. It can then be interpreted that cumulants of order greater than two measure the non-Gaussian nature of a time series [13]. Further, if a non-Gaussian signal is mixed with a Gaussian signal, higher-order cumulants will ignore the Gaussian noise portion of the signal [13].

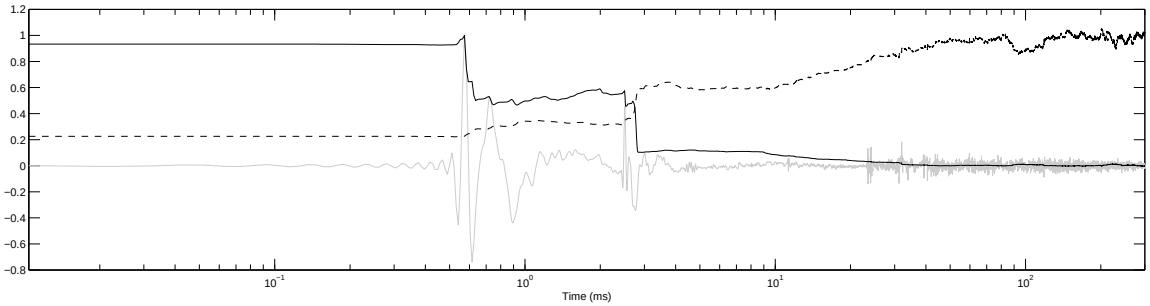
3.1. Kurtosis

The fourth order zero-lag cumulant of a zero-mean process is often referred to as kurtosis and can either be normalized or unnormalized [15]. Here kurtosis will refer to the normalized definition.

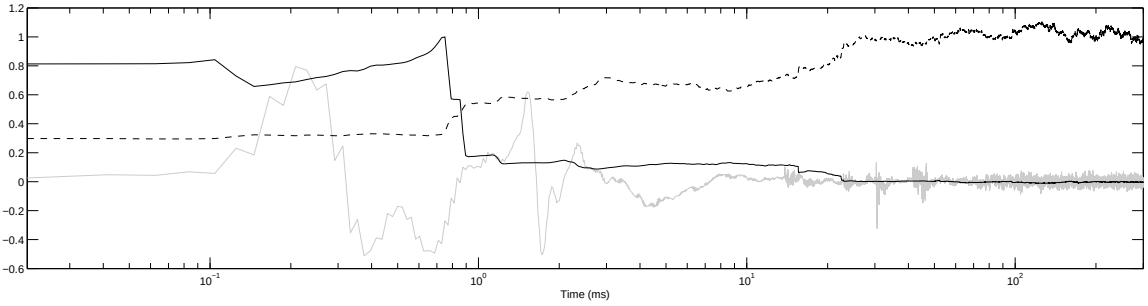
$$\gamma_4 = \frac{\mathbf{E}(x-\mu)^4}{\sigma^4} - 3 \tag{4}$$

where $\mathbf{E}()$ is the expectation operator, μ is the mean, and σ^2 is the standard deviation.

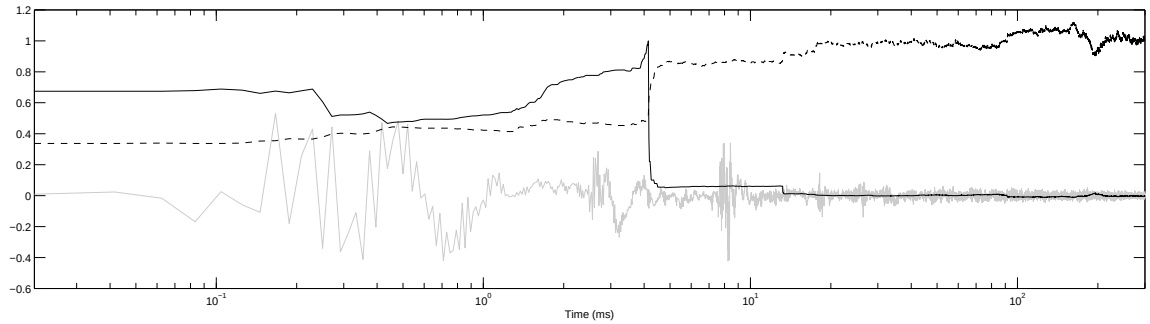
The same sliding window of 30 ms is used for the kurtosis analysis as for the standard deviation measurements. The values have been normalized to one within each IR merely for the sake of the figure. The values nearing zero, denoting a mixed room, and



(a) Sir Jack Lyons Concert Hall, York, UK, a 350 seat concert hall. Reverberation time T_{30} = 4.9 s



(b) York Minster in York, UK, a 330,000 m^3 church. Reverberation time T_{30} = 8.19 s.



(c) St. Andrew's in Lyddington, UK, a 2600 m^3 church. Reverberation time T_{30} = 1.5 s.

Figure 3: The solid gray lines are the IRs, the dashed lines are the normalized ratios of samples outside one standard deviation and the solid black lines are the normalized kurtosis values.

the rate of change of the values are more important than the absolute values, especially those of the peaks in the early reflections.

As can be seen in Fig. 3(a) to 3(c), the plots of the kurtosis show a distinct difference between the early and late energy unlike the plots of the standard deviation which only demonstrated the gradual transition. Two significant points can be seen on the plots: a rapid decrease in the kurtosis value and when the kurtosis value is first zero. All three spaces depicted in Fig. 3(a) to 3(c) have a large decrease in the kurtosis value at approximately the same time, ranging from 2 to 5 ms. Differences occur in the later point, when the kurtosis value is approximately zero. The time is much later in Fig. 3(a) at 45 ms; the larger, more reverberant spaces have earlier kurtosis values nears zero at 26 ms for Fig. 3(b) and 19 ms for Fig. 3(c).

4. FURTHER WORK

The ability to determine the transition time of an IR is useful for determining at what point in an IR an artificial reverberator can be used such as in [3]. This is particularly useful for hybrid reverberators such as in Fig. 1 which use a combination of convolution and filterbank reverberators.

While the kurtosis of an IR shows a significant difference in the early and late portions of a signal, the perceptual implications

of this transition have not been explored. Listening tests need to be conducted to determine if sufficient perceptual information is contained in the signal before the transition point. An IR truncated at the transition point needs to hold the same localization information as the complete IR. Only the reverberation tail which contains cues to the size of the space, not details of the localization of the sound source, should be removed.

There are a number of acoustical parameters such as reverberation time, clarity and early decay time that are dependent upon the absorption and diffusion of space. These measurements help characterize a space and quantitatively describe its suitability for music or speech. A study surveying these standard measurements and the transition point could be carried out to determine whether the transition time is useful measure of a space.

5. DISCUSSION

A measure that can determine the point in time when the early reflections have fully transitioned to late reverberation is described. Previous measures either need specific information about the space such as volume, or completely disregard the space and define the transition point to be 80 ms. A method that does not disregard the room properties but still does not need specific dimensions is implemented by analyzing the statistics of the IR of the space.

Higher order cumulants such as kurtosis are a more convenient measure for the transition time than lower order descriptive statistics. Kurtosis is essentially blind to symmetric probability distributions, unlike standard deviation. This gives a more definite threshold to determine a mixed room since the threshold when using standard deviation is dependent on the room.

Both standard deviation and kurtosis support the model of an IR being a deterministic system transitioning into a stochastic one, however they display this in two different ways. The standard deviation of the IR shows a gradual transition from the early reflections to late reverberation, but the earliest point in which the room is mixed is difficult to find. The kurtosis takes into account phase information while possessing blindness to symmetric distributions. A much sharper transition is then shown allowing a specific transition point to be easily selected.

The estimator in Eq.2 finds the York Minster to have a mixing time of 570 ms, while St. Andrew's is estimated to be mixed by 60 ms. Both are much later times than the kurtosis measurements of 26 and 19 ms respectively. The volume of the Sir Jack Lyon Concert Hall is not available, so the mixing time cannot be estimated from Eq.2.

The mixing time is relative to the beginning of the file containing the IR. In order for the calculated mixing time to be relevant amongst a group of IRs, the beginning of the files need to be standardized, preferably starting at the time of the impulse. Not all IRs contain the time between the impulse and the direct sound and some have the direct sound removed for use in convolution reverberators. This needs to be noted before multiple IRs are compared to each other.

Further work needs to be done to explore the perceptual relevance of this point in time and of the robustness of a truncated IR to still contain localization information. If listening tests show that an IR can be truncated to its transition point and still retain all sufficient localization information, then this method of truncation can be used in applications such as hybrid reverberators. The relationship between the transition time, other acoustical measures, and the properties of the room can also be studied further.

6. ACKNOWLEDGEMENTS

This research was supported by the Audio Engineering Society Educational Foundation and Queen Mary, University of London.

7. REFERENCES

[1] M.R. Schroeder, "Natural sounding artificial reverberation," *JAES*, vol. 10, no. 3, pp. 219–223, October 1962.

[2] James A. Moorer, "About this reverberation business," *Computer Music Journal*, vol. 2, Summer, 1979.

[3] Jean-Marc Jot, Laurent Cerveau, and Olivier Warusfel, "Analysis and synthesis of room reverberation based on a statistical time-frequency model," in *103rd AES Convention*, New York, NY, September 1997.

[4] Barry Blesser, "An interdisciplinary synthesis of reverberation viewpoints," *JAES*, vol. 49, no. 10, pp. 867–903, October 2001.

[5] Jonathan S. Abel and Patty Huang, "A simple, robust measure of reverberation echo density," in *121st AES Convention*, San Francisco, October 2006.

[6] H. Kuttruff, *Room acoustics*, Spon Press, London, 4th edition, 2000.

[7] Jean-Dominique Polack, "Playing billiards in the concert hall: the mathematical foundations of geometrical room acoustics," *Applied Acoustics*, vol. 38, pp. 235–244, 1993.

[8] ISO 3382, "Acoustics-measurements of the reverberation time of rooms with reference to other acoustical parameters," 1997.

[9] Durand R. Begault, *3-D sound for virtual reality and multimedia*, NASA, 2000.

[10] M.R. Schroeder, "Statistical parameters of the frequency response curves of large rooms," *JAES*, vol. 35, no. 5, pp. 307–316, May 1987.

[11] M.R. Schroeder, "Normal frequency and excitation statistics in rooms: model experiments with electric waves," *JAES*, vol. 35, no. 5, pp. 307–316, May 1987.

[12] Damian T. Murphy, "Archaeological acoustic space measurement for convolution reverberation and auralization applications," in *DAFX '06*, September 2006.

[13] Chrysostomos L. Nikias and Athina P. Petropulu, *Higher-order spectra analysis: a nonlinear signal processing framework*, Prentice Hall, 1993.

[14] Jerry M. Mendel, "Tutorial on higher-order statistics (spectra) in signal processing and system theory: theoretical results and some applications," *Proc. of the IEEE*, vol. 79, no. 3, pp. 278–305, March 1991.

[15] Ananthram Swami, Jerry M. Mendel, and Chrysostomos L. Nikias, *Higher-order spectral analysis toolbox*, MathWorks, 3rd edition, 1998.

AUTOMATIC MIXING: LIVE DOWNMIXING STEREO PANNER

Enrique Perez Gonzalez and Joshua Reiss

Centre for Digital Music,
Queen Mary University of London, Electronic Engineering,
Mile End Road, E1 4NS
London, United Kingdom
enrique.perez@elec.qmul.ac.uk
josh.reiss@elec.qmul.ac.uk

ABSTRACT

An automatic stereo panning algorithm intended for live multitrack downmixing has been researched. The algorithm uses spectral analysis to determine the panning position of sources. The method uses filter bank quantitative channel dependence, priority channel architecture and constrained rules to assign panning criteria. The algorithm attempts to minimize spectral masking by allocating similar spectra to different panning spaces. The algorithm has been implemented; results on its convergence, automatic panning space allocation, and left-right inter-channel phase relationship are presented.

1. INTRODUCTION

An audio engineer carefully handcrafts the characteristics of multiple inputs to downmix it into a constrained number of channels. Creating a mix involves numerous spectral and gain processing as well as the use of several audio effects. This research explores the automatisation of one of these processes. The spatial effect, which has been investigated, is a stereo panner algorithm. The panner under study downmixes *K* inputs and converts them into a stereo mix. This automatic stereo panner makes panning decisions based on constrained spectral rules as well as priority criteria. The algorithm has also been optimized for live downmixing situations where a second take is not an option.

The first automatic processing for live sound environments for mixing applications can be traced to Dugan's automatic microphone mixer [1, 2]. Dugan set the basic principles of automatic gain adjustment for automatic mixing. This type of mixer was able to maintain constant gain structure regardless of the number of active microphone inputs. This mixer was completely analog and based its decisions on time domain gain compensation. Several years later a mechanical approach based on directive sensitive gating for automatic mixing was developed by Julstrom [3]. Currently, in the authors' knowledge, no frequency domain approach to automatic mixing for live applications has been proposed. With current DSP processing power and the expanding availability of fully automated digital consoles, it should be possible to develop automatic processes for automatic mixing in an easy and cost-effective manner. Automatic mixing can prove useful in live mixing for video games, live concerts and post-production.

For the purpose of this research it is important to make a distinction between *automatic* mixing processes and *automated* mixing processes. An automatic process involves an autonomous process. This autonomous process can be treated as a constrained rule problem in which the design of the control rules determines the process to be applied to the input signals. The automated process, on the other hand, is the result of playing back in sequence a series of user recorded actions. This involves playing back previously recorded and stored actions, regardless of whether automatically or manually generated.

A common task in live mixing is downmixing a series of mono inputs into a two channel stereo mix. For doing this the input channels get summed into a Left (L) and a Right (R) channel. The proportion at which these multiple mono inputs are added to each L and R channels are responsible for the perceived stereo image. Previous related work on downmixing for spatial audio coding, from 5.1 surround to 2.0 stereo, has been attempted by [4]. Processing of multiple channels for real time applications using priority has been attempted by [5], but this method requires an off-line processing stage which requires pre-processing of the audio channel in order to enhance them with descriptors. This method is suitable for game and simulations but are not optimal for live environments where the signal nature is unknown. Work on upmixing has been researched by [6, 7]. In their work, they describe methods to turn a stereo downmix into a multi-channel upmix. Although these methods can prove useful if backtracked, they are more suitable for multi-channel surround format conversion rather than for multiple input mixing. By multiple input channels we refer to the individual instruments of a live group of musicians or multiple speech inputs as opposed to multi-channel format submixes, as contained in 5.1 surround formats. In the knowledge of the authors, no current approach to stereo downmixing multiple inputs channels in a live environment has been attempted.

Other relevant related work includes the idea of on-the-fly-mixing. On the fly multi-track mixing has as a central idea to maintain the intentions of the composer and sound engineer while providing the final user with some degree of control [8]. This system has the intention of enhancing the end user experience by providing him with controllable parameters, which have been constrained in order to keep even intention for non-expert user. The system proposed in this research differs in the idea that it searches an automatic approach to downmixing by reducing user interaction. The proposed system is to be seen as a helper to the sound engineer and composer rather than giving the engineer a new set of extra constraints parameters to be manipulated. The proposed approach seeks to enhance the user experience by automatically downmixing the input sources while reducing or eliminating the mixing tasks to the user. In particular, this paper ex-

plores an algorithm for downmixing using an automatic stereo panner.

2. AUTOMATIC PANNING

The panner is based on channel priority, the algorithm was implemented so that the user connects the most important input channels to the first channels of the downmixer and the least important channel inputs to the last channels of the downmixer. However, instrument recognition techniques can be used to implement an automatic version of channel priority. In either case channel priority gives the algorithm data, which can be used to help determine, with better accuracy, the panning positions. For example if the lead vocal is the most important channel in the mix, in the current implementation of the algorithm, it should be located on channel one. The algorithm will treat the highest priority channel as a source, which should be as centered as possible. The algorithm also is based on the assumption that panning channels which have similar spectral content to opposite sides improves intelligibility by reducing spectral masking. For the proposed algorithm the panning space is determined by the number of channel to be mixed and the available panning steps. Finally the algorithm has been set so that the panning step is dependent on the spectral resolution of the analysis. The algorithm has been optimized for MIDI control.

2.1. Design architecture

One of the most important design considerations for any automatic live sound process is the fact that input sources should be reproduced at all times. Missing just a few samples can mean losing important content of the live audio program to be heard. Figure 1 shows the general diagram of the proposed architecture that solves this problem.

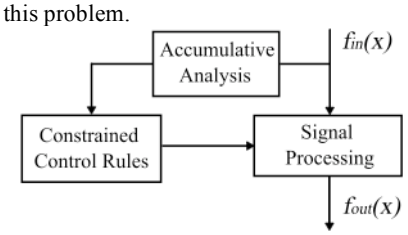


Figure 1: Diagrams of side processing algorithms for automatic live panning application.

Other important consideration is that, because of the live nature of the input signal an instantaneous decision based on the current sample can prove wrong. For this reason an accumulative knowledge based approach can prove more reliable. Because of this a side-chain accumulative analysis together with a constrained control rule structure is used to calculate the control parameters send to the desired signal processing procedure. This means that the input signal is free to reach the output even though the analysis and processing decisions have not been determined. Once the analysis and rules calculations have been finalized the processing can take place. This design gives the opportunity to the algorithm to constantly get updated by using the input data against past-accumulated data. This can influence the processing stage by using previous knowledge instead of performing decisions based on instantaneous transients, which could result in generating unwanted signal processing artifacts.

2.2. The filter bank analysis

The proposed system uses a filter bank as means of doing spectral analysis. The filter bank does not affect the output signal as it is only used for analysis. The filter bank uses first order band pass filters. The filters inside the bank are determined by the bandwidth contained between the -3dB low cut off and -3dB high cut off points of the filter. Each consecutive filter in the filter bank has double the bandwidth of the previous filter. The total number of filters inside the bank (K) is equal to the number of input channels to be mixed. This will result in an adaptive frequency analysis resolution, which is dependant on the number of input channels.

To design the channel dependent filter bank the following equations are provided.

$$Bw(k) = \frac{F_{\max}}{40K} 2^{k-1} \quad (1)$$

Where $Bw(k)$ correspond to the bandwidth of the k^{th} filter contained on the filter bank where k goes from 1 to K and K corresponds to the total number of input channels to be mixed, which is equal to the total number of filters in the filter bank. $Fmax$ is the maximum frequency to be reproduced and the constant 40 was chosen simply to give filter bandwidths at simple rational intervals.

$$LCF(k) = F_{\min}.(2)^{k-1} \quad (2)$$

Equation 2 calculates the Low Cut Frequency (LCF) for the k^{th} filter. Where k goes from 1 to K. $Fmin$ corresponds to the left -3dB cut off point of the first filter, in the current implementation $Fmin$ has a value of 20Hz. 20Hz has been chosen because it is widely accepted as the lower frequency limit of human hearing. To calculate the High Cut Frequency (HCF) for the k^{th} filter the corresponding $LCF(k)$ and $Bw(k)$ must be added together. Where k goes from 1 to K. Because K is equal to the total number of input channels to be mixed, we can say that k corresponds to the individual filter identifier number. The HCF corresponds to the right -3dB cut off point of the filter.

Once the filter bank has been designed the algorithm uses the spectral, band limited, information within each filter to obtain the absolute peak amplitude for each filter. The peak amplitude is measured within a 100ms window. Because noise contained in the signal of interest may trigger undesired readings a second threshold peak meter is used to gate the peak readings of each filter. During development it was found that the optimal threshold value for this application is -60dB, with a 10ms window. For every occurrence of the peak amplitude measurement that has is successful in passing through the gate, the following algorithm is performed.

First all K filters are scanned in search of the highest amplitude. Second all filters are searched with the aim to identify the k filter responsible for having the highest amplitude. Once this is done the result is accumulated in a register corresponding to the k^{th} filter responsible for the highest peak amplitude. Third the K accumulated registers are scanned in search of the k accumulator responsible for the highest number of occurrences. Finally the system outputs the k filter identifier number corresponding to the biggest accumulated spectral band of the input signal. This process is repeated in a continuous manner for every peak amplitude value received after the gating stage. This approach uses digital logic operations of comparison, equity and accumulation only,

which makes it highly attractive for an efficient digital implementation. The block diagram of the bank filter analysis algorithm has been sketched in Figure 2.

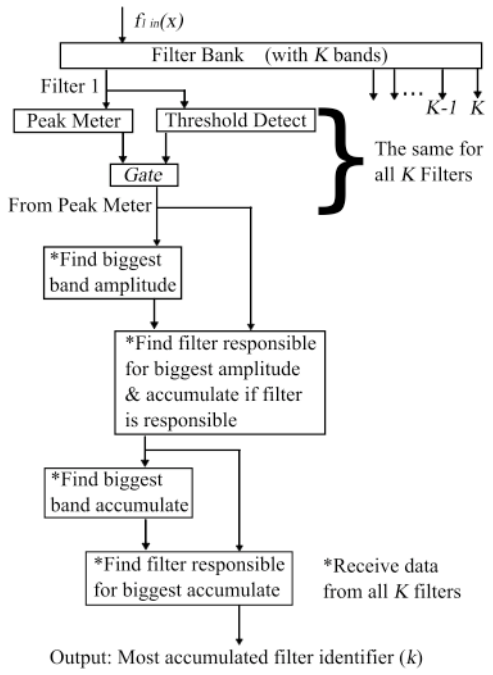


Figure 2: *Analysis block diagram for one input channel for a Filter bank of K length.*

2.3. The constrained control rules

The first rule consists of maintaining all sources whose main energy is contained in the lowest filter banks un-panned. There are two main reasons for doing this. First this ensures that the Low frequency content remains evenly distributed within speakers[9]. This ensures that reproduction of low frequencies, which are more likely to produce audible distortion at high levels, remains spread over two sources to reduce distortion. Second there is no point on panning low frequency sources below 200Hz because we fail to localize them properly [10]. It is thought that we are unable to localize properly low frequencies because its wavelength is so long that our left and right ears perceive them as coming simultaneously from the same place. It is also thought that because the head is unable to absorb the low frequencies, it is hard to localize them, in contrast, high frequencies do get some attenuation between ears which we associate with position.[11] For this reason all input signals, with accumulated energy contained in a filter with a HCF below 200Hz are not panned, and should remain centered at all times.

The second rule decides the available panning step *(PS)* a positioning rule based on equidistant spacing. The rule uses 2 parameters to determine its result. First the Integer location of the filter that contains the maximum number of accumulation *(k)*. Second the total number of occurrences of signals containing accumulated energy in the same k^{th} filter (R_k).

This PS has to be calculated for every different accumulated k^{th} filter. This means that for k^{th} filters which have not reached maximum accumulation there is no need to calculate the PS, this makes the algorithm less computationnally expensive. If only one repetition exists for a given k^{th} filter ($R_k=1$) the system is default to pan the input to the center (64 or 63 in the MIDI case). The system default initial value is to have all channels centered.

In order to obtain all the available panning space locations for R_k bigger than one equation 3 is provided.

$$PS_k(i) = RND[\frac{(i-1)\max PS}{R_k - 1}] \qquad (3)$$

Where $PS_k(i)$ is the i^{th} available panning step and i has a range from 1 to R_k. i and k have a range from 1 to K. Because the algorithm has been optimized for MIDI control the *maxPS* is 127, and the rounding function *RND* is to be used to obtain the proper discrete MIDI *PS*.

The third rule is priority-panning assignment. Priority is based on priority numbers (Pr_k) assigned to all R_k. In order to assign the panning priority first we must calculate PS with equation 3. An alternating priority algorithm has been used. This priority assignation works as follows; once K has been obtain by the algorithm, the system scans all channels to obtain the number of input channel sources which share the same accumulated filter R_k. Then the system proceeds to assign the priority channel number Pr_k from left to right. The Pr_k is done from left to right because it goes in accordance with the idea that the channel closest to the first mixer channel has more priority than the last one. This ensures that the channels closer to the first physical channels remain as centered as possible. This approach reduces spectral masking by separating inputs with similar spectral content as far as possible from each other. Since the implementation of this algorithm was intended for compatibility with MIDI controlled automated mixers the panning step has a range from 0 to 127 steps. Where centre is 64 or 63 and *maxPS*=127. In Table 1 it is presented all possible PS for an 8CH automatic panning system.

PS(i)	Pr_1	Pr_2	Pr_3	Pr_4	Pr_5	Pr_6	Pr_7	Pr_8
PS(1)	64		-	-	-	-	-	-
PS(2)	0	127	-	-	-	-	-	-
PS(3)	64	0	127	-	-	-	-	-
PS(4)	43	84	0	127	-	-	-	-
PS(5)	64	32	95	0	127	-	-	-
PS(6)	51	76	25	102	0	127	-	-
PS(7)	64	42	85	0	21	106	127	-
PS(8)	54	73	36	91	18	109	0	127

Table 1: *Discrete panning rule by implementing alternate priority*

All values have been obtained using equation 3 and by implementing alternate priority. It is important to notice that the maximum value of i is equal to the number of channels of the mixer, and the same applies for the maximum assignable priority numbers. Based on this, it can be stated that the total panning space can be calculated as the number of panning steps available times the filter bank filter which HCF are above 200Hz. It is also important to realise that thanks to this channel dependency the algorithm will update itself every time a new input is detected in a new channel. The algorithm will also update itself if the spectral content of a input channel suffers from a drastic change over time of spectral content. The block diagram containing the constrained decision control rule stage of the algorithm is presented in Figure 3.

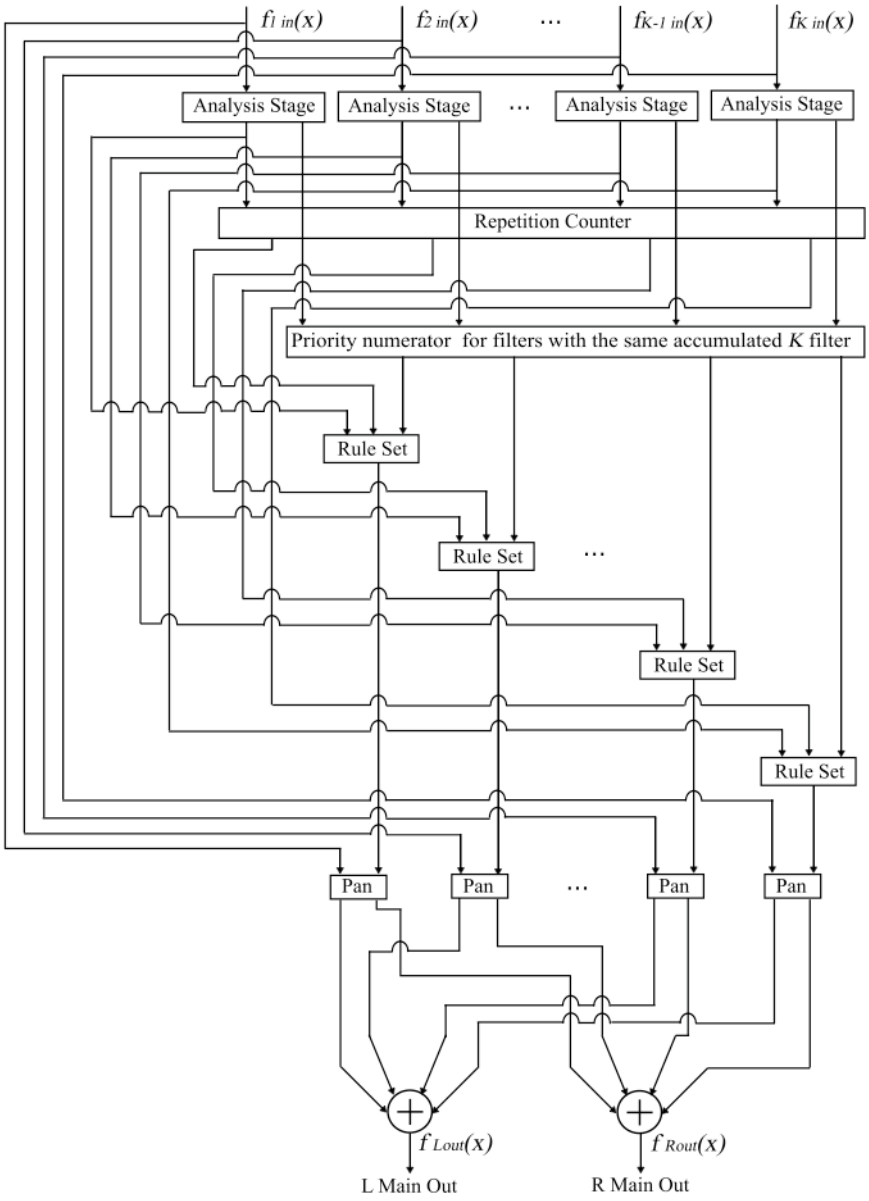


Figure 3: *Block diagram of Automatic Panner algorithm constrain control rules algorithm for K input Channels*

2.4. The panning processing

The panning architecture used consists of a -3dB panning law. This means that when panning is centred the left (L) and right (R) channel have a -3dB gain. This is so, that when L and R channels are in phase and added, a 0dB gain is achieved. When the panner goes all the way to the left the L channel has a normalized gain of 0dB and the R channel has a gain of -∞, and the inverse when the system is panned to the R channel. This approach keeps a constant 0dB gain regardless of the panning angle. The panning angle is the degree measurement of rotation of a pan-pot. The panning step is the digital analogy used for MIDI implementation of a panning angle. The panning step is related to the signal gain multiplier. The panning algorithm uses the following equations for processing the right and left channel.

$$f_{Rout}(x) = (PS_k(i) / \max PS).f_{in}(x) \quad (4)$$

$$f_{Lout}(x) = [1 - (PS_k(i) / \max PS)].f_{in}(x) \quad (5)$$

The term multiplying $f_{in}(x)$ in equations 4 and 5 have a range from 0 to 1 and it is called the Panning Factor (PF). It has a maximum range of 1 In order to maintain the resulting signal normalized. The PF of one channel is complementary to the PF of the other. For this reason the sum of the squares of boat PF is always 0dB, making the overall amplitude equal to the original amplitude of $f_{in}(x)$ regardless of the panning position.

An interpolation algorithm has been coded into the panner to avoid rapid changes of signal level. The interpolator has a 22ms fade-in and fade-out, which ensures a smooth natural transition when the panning control step is changed.

3. RESULTS

Several sinusoidal test signals and music tracks simulating a live playing band were used as a mean to test the automatic panning algorithm. The multi-track data used was obtained from the BASS-dB database [12]. BASS-dB is the Blind Audio Source Separation evaluation database; it contains links to multi-track recordings which license allows modification and redistribution of the data for non-commercial purposes.

In all studied cases the algorithm was able to converge. In Figure 4 we can see the convergence for 4 different sources. The 4 sources were selected from a set of measurements obtained from an 8ch automatic panning downmixer. The plot shows the panning factor as it approaches stable state, as applied to the input signal.

The dotted line, in Figure 4, is the result of plotting the PF of one of 4 tracks that have similar spectral content; the algorithm has spread all four signals equidistantly. The dotted line corresponds to the highest priority channel out of the four tracks. The dotted line has converged into a panning factor equal to 0.33858 or a MIDI panning step of 43. The other 3 sources not shown in this plot converged in accordance to Table 1 for a *PS(4)*.

It is important to notice that the speed at which the algorithm converges is dependant on the spectral content of the overall input channels. Also a track containing similar spectral content which start later in time than others can cause a panning space re-assignation. Others convergence values where the source has been panned fully to the sides have been also plotted (PF=0 and PF =1). Finally the solid line represents a drum kit signal, which although the algorithm struggles to decide wheatear its spectral content is of mainly high frequencies (due to the hi-hat) or low frequencies (due to the kick drum) it manages to converge into central position (PF=0.5), which is technically the most convenient panning position for a signal containing very low frequencies.

In Figure 5 we can see the panning factor depicted as a solid line in Figure 4 superimposed on the MIDI panning step calculated by the algorithm. The MIDI panning step is one stage before the 2000 sample interpolation is applied to the panning process. These results show how the interpolation step makes the automatic panner more resilient to panning positioning flutter while achieving a more natural pan.

In Figure 6 the result of downmixing 12 sinusoidal test signals through the automatic panner are shown. It can be seen that both *f1* and *f12* are kept centered and added together because their spectral content is below 200Hz. The three sinusoids with a frequency of 5KHz have been evenly spread. *F2* has been allocated to the center due to priority; while *f4* has been send to the left and *f6* has been send to the right, in accordance to Table 1 for a *PS(3)*. Because there is no other signal with the same spectral content than *f11* it has been assigned to the center. The four sinusoids with a spectral content of 15Khz have been evenly spread. Because of priority *f3* has been assigned a MIDI step of 43, *f7* has been assigned a MIDI step of 84 steps, *f9* has been assigned all the way to the left, and *f10* has been assigned all the way to the right, in accordance to Table 1 for a *PS(4)*. Finally the two sinusoids with a spectral content of 20KHz have been panned to opposite sides. *f5* has been send to the left while *f8* has been send to the right, in accordance to Table 1 for a *PS(2)*. All results prove to be in accordance to the constrained rule equations proposed in section 2.3.

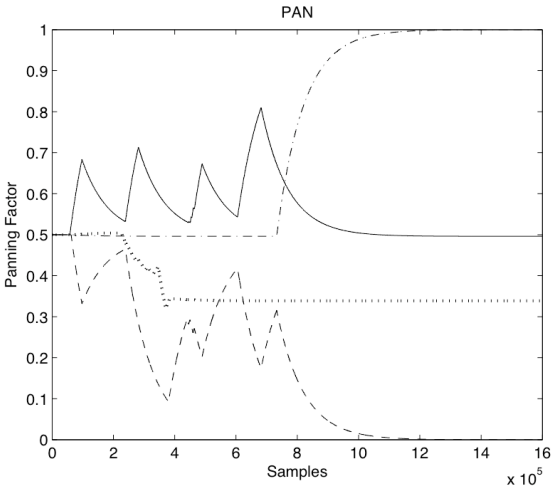


Figure 4: *Convergence of automatic panning algorithm for 4 different convergence values. (-) Panning Factor for a drum kit track,(--)panning Factor for a bass guitar, (.-) panning Factor for a vocal track, and (..) panning Factor for a channel input which spectral content is concentrated in the same filter.*

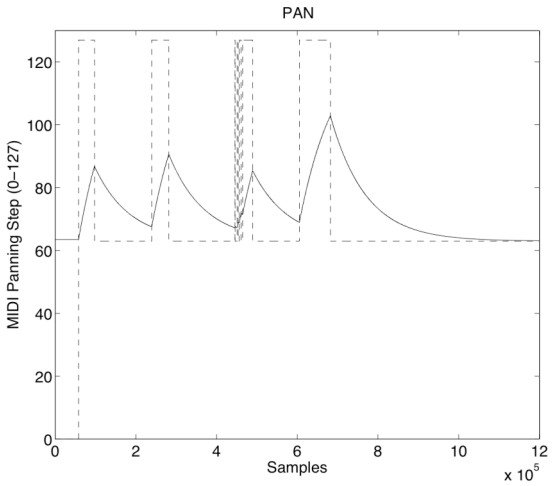


Figure 5: *Discrete panning step (- -). Super imposed interpolative panner angle (-) as applied to an input signal consisting of a drum kit recording.*

A Lissajous curve or stereogram is a two dimensional representation of a stereophonic signal and is usually perform by using a oscilloscope in XY mode or by using a vector oscilloscope. The stereogram can be obtained by plotting in time-synchronicity the left channel against the right channel. This measurement provides detailed information concerning inter-channel phase relationship [13]. The data contained in the stereogram of Figure 7 is widely spread in an oval. This means that the phase relation between the left and right channel is close to 90deg. This means we have achieved a wide spread stereo signal. In order to have a reference, a mono signal, which is represented by the diagonal separating the left and right planes of the stereogram has been plotted. The

plot also shows a good data equilibrium between the right and left channel.

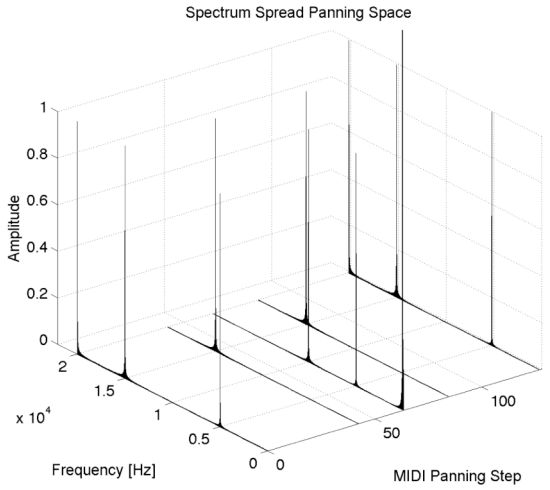


Figure 6: *Spectrum spread panning space for a 12 input CH automatic panner based on the proposed design. The test inputs were sinusoids with amplitude equal to one and the following frequencies: f1*=125Hz, *f2*=5KHz, *f3*=15KHz, *f4*=5KHz, *f5*=20KHz, *f6*=5KHz, *f7*=15KHz, *f8*=20KHz, *f9*=15KHz, *f10*=15KHz, *f11*=10KHz, and *f12*=125 Hz.

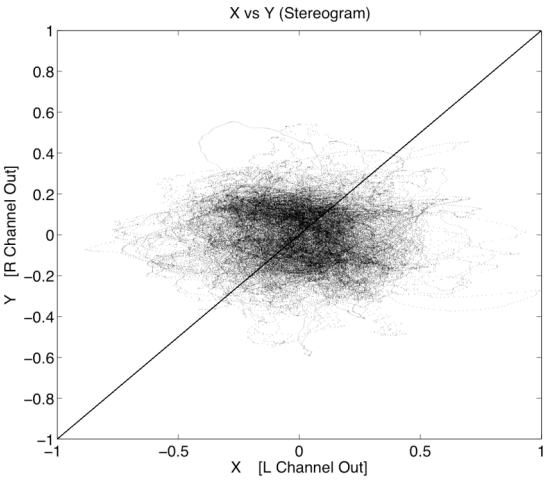


Figure 7: *Stereogram of 100,000 samples of a 5CH automatic panner. The samples correspond to a section in time where all 5-channel instruments are interacting simultaneously.*

4. CONCLUSIONS

An automatic panning algorithm for live multi-track sources has been successfully implemented. The algorithm reduces spectral masking while achieving a wide stereo signal. The system achieves panning by using constrained rules and bank filter accumulative techniques. Several test signals and multi-track signals have been tested. Results on its convergence, automatic panning space allocation, and stereogram have been presented. Subjective testing needs to be done in order to optimize the constrained rules and to verify the validity of our assumptions concerning placement of instruments and avoidance of spectral masking. Further work will also use instrument recognition techniques to automate priority rules that are independent of user channel connections. Filter bank optimization using psycho acoustical optimized band pass filters remains to be researched.

5. REFERENCES

[1] D. Dugan, "Automatic Microphone Mixing," in *51st Convention Audio Engineering Society*, San Fransico, California, 1975.

[2] D. Dugan, "Application of Automatic Mixing Techniques to Audio Consoles," in *87th Convention Audio Engineering Society* New York, 1989.

[3] S. Julstrom, and T. Tichy, "Direction-Sensitive Gating: A New Approach to Automatic Mixing," *Audio Engineering Society,* vol. 32, pp. 490-506, Julay/August 1984.

[4] B. Schick, R. Maillard, and C-C. Spenger, "First investigations on the use of manually and automatically generated stereo downmixes for spatial audio coding," in *118th Convention Audio Engineering Society* Barcelona, Spain, 2005.

[5] T. Nicolas, "Scalable Perceptual Mixing and Filtering of Audio Signals Using an Augmented Spectral Representation," in *DAFx' 05* Sophia Antipolis, France, 2005.

[6] C. Advendano, and J.-M. Jot, "A frequency-Domain Approach to Multichannel Upmix," *Journal Audio Engineering Society,* vol. 52, pp. 740-749, 2004.

[7] Y. Li, and P. F. Driessen, "An Unsupervised Adaptive Filtering Approach of 2-To-5 Channel Upmix," in *119th Convention Audio Engeneering Society* New York, 2005.

[8] F. Pachet, and O. Delerue, "On-the-Fly Multi-Track Mixing," in *109th Convention Audio Engineering Society* Los Angeles, California, USA, 2000.

[9] P. White, "Pan Position," in *The Sound on Sound book od desktop digital sound*., M. Books, Ed., 2000, pp. 169-170.

[10] E. Benjamin, "An Experimental Verification of Localization in Two-Channel Stereo," in *121st Convention Audio Engineering Society* San Fransisco, California, USA, 2006.

[11] J. Beament, "The direction-finding system," in *How we Hear Music, The relationship Between Music and the Hearing Mechanism*, T. B. Press, Ed. Suffolk UK, 2001, pp. 127-130.

[12] E. Vincent, R. Gibonval, C. Favotte, and al., "BASS-dB: The Blind Separation Audio Source Separation Evaluation Database, http://www.irisa.fr/metiss/BASS-dB/," 2006.

[13] E. B. Brixen, "Audio Levels and Readings, http://www.dk-technologies.com/downloads/Audio%20Levels.pdf," in *Addition to User's manual of The Master Stereo Displays from DK-Audio*: DK-Audio, 2007.

ADAPTIVE HARMONIZATION AND PITCH CORRECTION OF POLYPHONIC AUDIO USING SPECTRAL CLUSTERING

Mathieu Lagrange, Graham Percival, and George Tzanetakis

Computer Science Dept.
University of Victoria, British Columbia
Canada
[lagrange, gperciva, gtzan]@uvic.ca

ABSTRACT

There are several well known harmonization and pitch correction techniques that can be applied to monophonic sound sources. They are based on automatic pitch detection and frequency shifting without time stretching. In many applications it is desired to apply such effects on the dominant melodic instrument of a polyphonic audio mixture. However, applying them directly to the mixture results in artifacts, and automatic pitch detection becomes unreliable. In this paper we describe how a dominant melody separation method based on spectral clustering of sinusoidal peaks can be used for adaptive harmonization and pitch correction in mono polyphonic audio mixtures. Motivating examples from a violin tutoring perspective as well as modifying the saxophone melody of an old jazz mono recording are presented.

1. INTRODUCTION

Pitch correction and harmonization are some of the most common digital audio manipulations. They are typically applied to recordings of monophonic sound sources such as the singing voice or melodic instruments. Currently if the original monophonic sound source is not available it is not possible to apply these effects on polyphonic audio mixtures. The main problem is that the required frequency shifting is also applied to the accompaniment, background music resulting in severe artifacts. In addition, pitch correction and harmonization are adaptive effects that rely on the output of an automatic pitch detection algorithm. In general pitch detection of the dominant melody in polyphonic audio is much harder than the monophonic case and in most cases unreliable.

In this paper we describe how a dominant melody separation algorithm inspired by ideas in Computational Auditory Scene Analysis (CASA) can be utilized for applying pitch corrections and harmonizations to the melody in polyphonic audio recordings. Unlike systems that use stereo panning information [1] our focus is mono recordings where there is a clear dominant melody such as a violin with piano accompaniment or saxophone and trumpet melodies in jazz recordings. The source separation algorithm is based on a sinusoidal representation and a spectral clustering technique is used to group the peaks of the dominant melody. Perceptually informed grouping cues such as amplitude/frequency proximity and harmonicity are utilized. Pitch correction and harmonization are straightforward to express using a sinusoidal representation by simple frequency scaling of the peaks. This assumes that the underlying fundamental frequency of the dominant melody is known. We utilize a pitch detection algorithm based on [2] and show that it works better using the separated signals.

The remainder of the paper is structured as follows. The dominant melody separation algorithm is described in section 2. Pitch correction and harmonization are described in section 3. Experiments with a violin tutoring system and altering the saxophone melody in a jazz recording are described in section 4. The paper ends with conclusions and directions for future work.

2. SOUND SOURCE FORMATION

Computational Auditory Scene Analysis (CASA) systems aim at identifying perceived sound sources (e.g. notes in the case of music recordings) and grouping them into auditory streams using psycho-acoustical cues [3]. However, as remarked in [4] the precedence rules and the relevance of each of those cues with respect to a given practical task is hard to assess. Our goal is to provide a flexible framework where these perceptual cues can be expressed in terms of similarity between time-frequency components. The identification task is then carried out by clustering components which are close in the similarity space. Therefore, the complexity of the algorithm is strongly related to the number of components considered. In this paper we use time-varying sinusoids as the underlying time-frequency representation.

2.1. Sinusoidal Modeling

Most CASA approaches consider auditory filterbanks and/or correlograms as their front-end [5]. In these approaches the number of time-frequency components is relatively small. However closely-spaced components within the same critical band are hard to separate. Other approaches [4, 6, 7] consider the Fourier Spectrum as their front-end. In these approaches, in order to obtain sufficient frequency resolution a large number of components is required. Components within the same frequency region can be pre-clustered together according to a stability criterion computed using statistics over the considered region. However, this approach has the drawback of introducing another clustering step, and opens the issue of choosing the right descriptors for those pre-clusters. Alternatively, a sinusoidal front-end is helpful to provide meaningful and precise information about the auditory scene while considering only a limited number of components, see Figure 1, and is the representation we consider in this work.

Sinusoidal modeling aims to represent a sound signal as a sum of sinusoids characterized by amplitudes, frequencies, and phases. A common approach is to segment the signal into successive frames of small duration so that the stationarity assumption is met. For each frame a set of sinusoidal components is estimated.

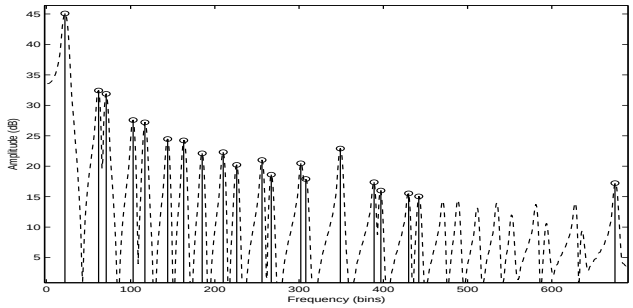


Figure 1: *Picking of 20 peaks in the spectrum of a mixture of two harmonic sources.*

The discrete signal $x^k(n)$ at frame index k is then modeled as follows:

$$x^k(n) \quad = \quad \sum_{l=1}^{L^k} a_l^k \cos\left(\frac{2\pi}{F_s} f_l^k \cdot n + \phi_l^k\right) \qquad (1)$$

where F_s is the sampling frequency and ϕ_l^k is the phase at the beginning of the frame of the l-th component of L^k sine waves. The f_l and a_l are the frequency and the amplitude of the l-th sine wave, respectively, both of which are considered constant within the frame. For each frame k, a set of sinusoidal parameters $\mathcal{S}^k = \{p_1^k, \cdots, p_{L^k}^k\}$ is estimated. The system parameters of this Short-Term Sinusoidal (STS) model $\mathcal{S}^k$ are the L^k triplets $p_l^k = \{f_l^k, a_l^k, \phi_l^k\}$, often called *peaks*.

These parameters can be efficiently estimated by picking some local maxima from a Short-Term Fourier Transform (STFT) with a frame size of 46ms and a hop size of 11ms. The precision of these estimates is further improved using phase-based frequency estimators which utilize the relationship between phases of successive frames [8]. Using this enhanced frequency estimate, the rough amplitude estimate provided by the magnitude of the local maximum is also corrected.

2.2. Spectral Clustering

In order to simultaneously optimize partial tracking and source formation, we construct a graph over the entire duration of the sound mixture of interest. Unlike approaches based on local information [9], we utilize the global normalized cut criterion to partition the graph (spectral clustering). This criterion as been successfully used for image and video segmentation [10]. In our perspective, each partition is a set of peaks that are grouped together such that the similarity within the partition is maximized and the dissimilarity between different partitions is maximized. By appropriately defining the similarity between peaks, a variety of perceptual grouping cues can be used.

To express such similarity, the edge weight connecting two peaks p_l^k and $p_{l'}^{k'}$ (k is the frame index and l is the peak index) may depend on the proximity of frequency, amplitude and harmonicity:

$$W(p_l^k, p_{l'}^{k'}) = W_f(p_l^k, p_{l'}^{k'}) \cdot W_a(p_l^k, p_{l'}^{k'}) \cdot W_h(p_l^k, p_{l'}^{k'}) \qquad (2)$$

where W_x are typically radial basis functions of distance among the two peaks in the x axis. For more details see [11, 12].

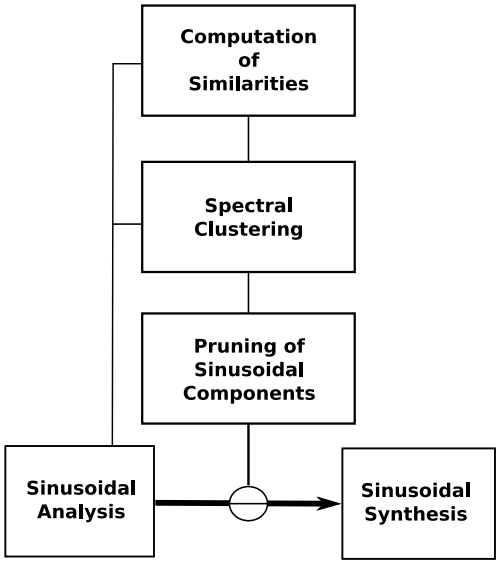


Figure 2: *Block-Diagram of the Dominant Melody Segregation Algorithm.*

Most existing approaches that apply the Ncut algorithm to audio [13, 7] consider the clustering of components over one analysis frame only. However, the time integration (partial tracking) is as important as the frequency one (source formation) and should be carried out at the same time. We therefore propose in [12] to consider the sinusoidal components extracted within a texture window of several spectral frames (20 in the experiments). Figure 2 shows a block diagram of the proposed separation scheme.

We considered a maximum of 40 sinusoids per frame. Those frames are 20 ms long. We chose to select two out of three clusters of peaks for each texture window to perform the resynthesis. The clusters with the highest average within similarity based on Equation 2 are selected. An example of separation is depicted in Figure 3. A bank of sinusoidal oscillators is used for resynthesis.

3. PITCH CORRECTION AND HARMONIZATION

Pitch correction and harmonization both rely on the ability to modify the perceived pitch of a sound without changing its duration in time. For the sounds we are interested in, the pitch directly corresponds to the fundamental frequency of the sound, therefore we do not differentiate between pitch and fundamental frequency. A sinusoidal representation is ideally suited for this purpose and therefore most systems for pitch modification and harmonization are based on the phasevocoder [14]. These effects work particularly well for signals with slowly varying harmonic structure that have small amounts of transients and background noise.

Using a sinusoidal representation, pitch shifting is achieved by scaling the frequencies of each peak in the representation. In the case of pitch correction we multiply the peaks corresponding to the dominant melody and do not retain the original peaks. For harmonization both the original and the multiplied peaks are retained for resynthesis and multiple scaling factors are used to form chords.

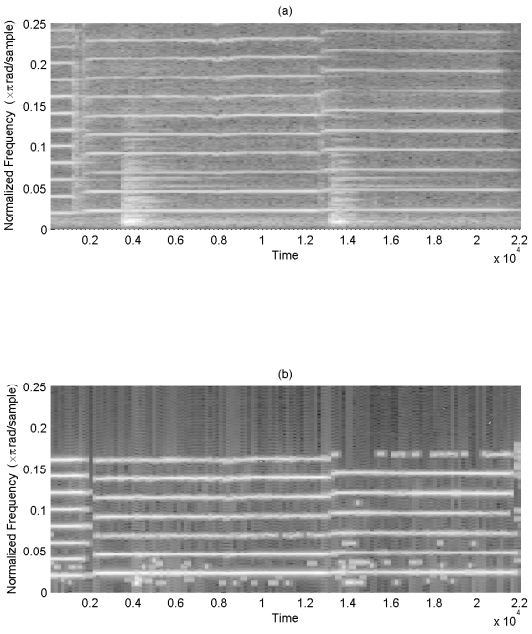


Figure 3: *Example of Segregation of the Violin (b) from a Violin accompanied by Piano.*

3.1. Pitch Detection

In order to know the required amount of frequency scaling it is necessary to know the underlying pitch of the sound during the frame under consideration. To automatically determine the pitch an implementation of the method described in [2] was used. The estimation of the pitch from a monophonic signal is a well studied area and robust methods exist, see [5] for a review.

We utilize the autocorrelation function to efficiently estimate the fundamental frequency ($f0$). For a time signal $s(n)$ that is stationary, the autocorrelation $r_s(\tau)$ as a function of the lag τ is defined as

$$r_s(\tau) = 1/N \sum_{j=t}^{t+N} s(t)s(t+\tau) \quad (3)$$

This function has a global maximum for $\tau = 0$. If there are also additional global maxima, the signal is called periodic and there exists a lag τ_0, the period, so that all these maxima are placed at the lags $n\tau_0$, for every integer n, with $r_s(n\tau_0) = r_s(0)$.

The inverse of the lag τ_0 provides an estimation of the fundamental frequency $f0$. The period is determined by scanning $r_t(\tau)$, starting at zero, and stopping at the first global maximum with non-zero abscissa. Quadratic interpolation is used to further improve the frequency estimation. In practical cases, the relative amplitude of those maxima may change and some others maxima may appear due to small aperiodicities of the signal. The issue is then to relevantly select which maximum corresponds to the $f0$ by considering several candidates under a plausible range and pick the one with the highest confidence, see [2, 15] for further references on the algorithm. To avoid picking harmonics as the fundamental frequency the amplitude of the peaks is weighted using the following equation:

$$r_s(\tau) = r(\tau_{max}) - OctaveCost^2 * log(MinPitch * \tau_{max}) \quad (4)$$

3.2. Note Segmentation

After the pitch contour has been generated we apply note segmentation to convert the frequency values in Hz to floating-point MIDI note numbers. The contour is scanned with a window size of 500ms. If the median frequency value is sufficiently far away from the previously detected notes (we found that 0.6 MIDI notes produced good results), the sample is flagged as a note boundary.

Once we have divided the audio into notes, we compare the pitches with the pitches given by a MIDI file. For each audio frame inside each note, we compute

$$frequency\_multiplier = \frac{midi2Hz(expected\_pitch)}{midi2Hz(detected\_pitch)} \quad (5)$$

It would be desirable to previously estimate the tuning of the performance. This way, if a different tuning standard (A = 442 Hz instead of 440 Hz, for example) is desired, we can easily change the mapping from Hz to floating-point values.

Currently, if a vibrato is present in the performance, the frequency contour is linearized within each note. Removing any micro-modulations of the pitch contour is desirable for correcting the performances of inexperienced students. However, this is not desirable for performances of skilled musicians who deliberately produce such modulations (vibrato, glissandi, ...).

3.3. Harmonization

Adding chords to a melody is a standard task in Computer Assisted Composition (CAC) [16], but this generally requires an analysis of the entire piece of music and is not suitable for online processing. In some cases we can use simple online algorithms to add chords. The easiest to automatically add harmonies is to have them precomputed by a human: "at 1.0 seconds, play a C chord. At 1.5 seconds, play a G7 chord, etc". This requires that we know exactly what pitch the user should be creating at every moment.

While this is unreasonable for an improvised performance of contemporary computer music, there are still situations where this constraint is quite valid. In music pedagogy, we often ask students to play their instrument in time with a metronome – in other words, to play specific notes at exactly the right time. This is also true of karaoke – since we are playing a vocal-less version of the music, we know what the user should be singing at every moment of the song. In these cases, adding pre-composed harmonies based purely on the elapsed time is an entirely appropriate method.

In this paper we use a simple and naive harmonization algorithm as a proof-of-concept. Simple harmonies may be added by examining only the current pitch. If we know that the user will be playing a simple melody in a specific key, we can calculate the scale degree of each note. "Scale degree" is the musical term for number of semitones within the octave. Adequate – although not particularly interesting – harmonies can be created by examining only the scale degree. For this paper, we used this simple method to demonstrate that harmonization is possible. More sophisticated approaches can easily be added to the system but are beyond the scope of this paper.

	violin solo	violin + piano	violin sep.
good intonation	97.9%	83.5%	93.1%
bad intonation	93.1%	79.9%	93.0%

Table 1: *Comparison of pitch recognition for separated and non-separated audio.*

4. EXPERIMENTS

In order to demonstrate how our proposed systems works we show examples from two application areas: pitch correction in music pedagogy and modifying the saxophone melody in monophonic jazz recordings. Audio examples can be found at `http://opihi.cs.uvic.ca/Dafx2007/`

Students learning instruments without fixed pitch (*i.e.* violin, cello, trombone) spend a significant amount of time concentrating on their intonation. Music teachers may demonstrate sections for the students, or encourage students to listen to recordings, but the sound of an expert playing an expensive musical instrument is quite different from the sound of a beginner playing a cheap musical instrument. Students may compare their sound to the expert's sound and become discouraged or distracted from the current goal of correct intonation. On the other hand, playing the correct melody using MIDI samples results in poor sound quality.

Using the student's own sound avoids these problems: the student hears music as it would sound if she played it with correct intonation. This eliminates any doubt (or possible excuses) for the student: the student listening to their pitch-corrected sound knows that she should – and *can* – produce exactly the same sounds. The audio examples are based on the following scenario: a violin student practicing with a tutoring system with piano accompaniment. A standard laptop microphone is used for acquisition and the recording is noisy. Various examples of pitch correction and harmonization for this scenario can be found on the webpage.

We also achieved significantly better pitch detection (and therefore better note segmentation) by separating the dominant melody from the original mixed audio. Table 1 shows our results; any pitch that was within 1.0 MIDI note of the real value was deemed to be "correct".

Another example application is the pitch correction of melodies in old mono recordings that are either live or for which the original master tapes of the individual instruments are not available. In the example provided on the webpage we modify the saxophone melody of a well known jazz standard by pitch shifting certain notes in the separated signal and then remixing with the residual. The result sounds quite good and to some extent the resynthesis artifacts are masked by the addition of the residual background. We are currently working on improving the resynthesis quality.

5. CONCLUSIONS

Dominant melody separation using a spectral clustering approach over a sinusoidal representation can be used for adaptive pitch correction and harmonization of polyphonic audio recordings.

Directions for future work include: incorporating more cues into the peak similarity calculation, improving resynthesis by reducing artifacts, more sophisticated harmonization, and adding the pitch correction functionality to an open source audio editing environment such as Audacity.

6. REFERENCES

[1] C. Avendano, "Frequency-domain source identification and manipulation in stereo mixes for enhancement, suppression and re-panning applications," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2003.

[2] P. Boersma, "Accurate short-term analysis of the fundamental frequency and the harmonics-to-noise ratio of a sampled sound," in *Proceedings of the Institute of Phonetic Sciences, Amsterdam*, 1993, vol. 17, pp. 97–110.

[3] A.S. Bregman, *Auditory Scene Analysis: The Perceptual Organization of Sound*, MIT Press, 1990.

[4] E. Vincent, "Musical source separation using time-frequency priors," *IEEE Trans. on Audio, Speech and Language Processing*, vol. 14(1), pp. 91–98, 2006.

[5] D. Wang and G. J. Brown, Eds., *Computational Auditory Scene Analysis: Principles, Algorithms and Applications*, Wiley, 2006.

[6] S.H. Srinivasan, "Auditory blobs," in *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, 2004.

[7] F.R. Bach and M. I. Jordan, "Blind one-microphone speech separation: A spectral learning approach," in *Proc. Neural Inf. Processing Systems (NIPS)*, Vancouver, Canada, 2004.

[8] M. Lagrange and S. Marchand, "Estimating the instantaneous frequency of sinusoidal components using phase-based methods," *to appear in the Journal of the Audio Engineering Society*, 2007.

[9] R.J. McAulay and T.F. Quatieri, "Speech analysis/synthesis based on sinusoidal representation," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. 34(4), pp. 744–754, 1986.

[10] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 22(8), pp. 888–905, 2000.

[11] M. Lagrange and G. Tzanetakis, "Sound source tracking and formation using normalized cuts," in *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, Honolulu, USA, 2007.

[12] Mathieu Lagrange, Luis Gustavo Martins, Jennifer Murdoch, and George Tzanetakis, "Normalized cuts for singing voice separation and melody extraction," *submitted to the IEEE Trans. on Acoustics, Speech, and Signal Processing (Special Issue on Music Information Retrieval)*.

[13] S.H. Srinivasan and M. Kankanhalli, "Harmonicity and dynamics based audio separation," in *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, 2003.

[14] J. Laroche and M. Dolson, "New phase-vocoder techniques for real-time pitch shifting, chorusing, harmonizing, and other exotic audio modifications," *J. Audio Eng. Soc*, vol. 47, no. 11, 1999.

[15] P. Boersma and D. Weenink, "Praat: doing phonetics by computer (version 4.5.06)," Retrieved December 13, 2006, from http://www.praat.org/.

[16] G. Assayag, "Computer assisted composition today," in *1st Symposium on Music and Computer, "Applications on Contemporar y Music Creation, Esthetic and Technical Aspects"*, 1998.

MODAL DISTRIBUTION SYNTHESIS FROM SUB-SAMPLED AUTOCORRELATION FUNCTION

Thomas Lysaght
Department of Computer Science, NUI, Maynooth
Co. Kildare, Ireland
Tom.Lysaght@nuim.ie

Joseph Timoney
Department of Computer Science, NUI, Maynooth
Co. Kildare, Ireland
Joseph.Timoney@nuim.ie

Victor Lazzarini
Department of Music, NUI, Maynooth
Co. Kildare, Ireland
Victor.Lazzarini@nuim.ie

ABSTRACT

The problem of signal synthesis from bilinear time-frequency representations such as the Wigner distribution has been investigated [1,2,4] using methods which exploit an outer-product interpretation of these distributions. The Modal distribution is a time-frequency distribution specifically designed to model the quasi-harmonic, multi-sinusoidal, nature of music signals and belongs to the Cohen general class of time-frequency distributions. Existing methods of synthesis from the Modal distribution [3] are based on a sinusoidal-analysis-synthesis procedure using estimates of instantaneous frequency and amplitude values. In this paper we develop an innovative synthesis procedure for the Modal distribution based on the outer-product interpretation of bilinear time-frequency distributions. We also propose a streaming object-oriented implementation of the resynthesis in the SndObj library [6] based on previous work which implemented a streaming implementation of the Modal distribution [7]. The theoretical background to the Modal distribution and to signal synthesis of Wigner distributions is first outlined followed by an explanation of the design and implementation of the Modal distribution synthesis. Suggestions for future extensions to the synthesis procedure are given.

1. INTRODUCTION

The Modal distribution was introduced by Pielemeier and Wakefield [3] as a member of the Cohen general class of time-frequency distributions [5] for the analysis of music signals. It is primarily a Wigner distribution, or more specifically, a smoothed pseudo-Wigner distribution (SPWD), with a kernel that takes account of the *modes* present in quasi-harmonic, multi-sinusoidal, music signals. Being based on the Wigner distribution, it provides a more accurate measure of time-frequency localisation and does not suffer from the time-bandwidth trade-off inherent in the spectrogram (also a member of the Cohen class) implementations. One drawback of the Wigner distribution is the existence of cross-terms amounting to beats between partials not existing in the original signal. The Modal distribution kernel is designed to minimize the effect of these cross terms for music signals. Furthermore, implementation of the time-smoothing kernel for the Modal distribution greatly reduces the number of Digital Fourier Transforms (DFTs) that need to be performed on the smoothed autocorrelation function and results in applying the DFT at hop steps related to the size of the time-smoothing kernel. Ultimately this decreases the load in computing the distribution. In order to apply an outer-product based synthesis procedure to the Modal distribution, therefore, it is necessary to devise a method of signal recovery from sub-sampled autocorrelation functions.

2. THEORETICAL BACKGROUND

Leon Cohen [5] proposed a general class of time-frequency distributions which are related through linear transformations. The set of all linear transformations of the Wigner distribution has come to be known as the Cohen general class. A two-dimensional kernel determines the linear transformation involved. The Wigner distribution, equation (1), in terms of the signal $f(t)$ and the spectrum $F(\omega)$ is given by:

$$W(t,\omega)=\frac{1}{2\pi}\int f^*\left(t-\frac{1}{2}\tau\right)f\left(t+\frac{1}{2}\tau\right)e^{-j\tau\omega}d\tau = \frac{1}{2\pi}\int F^*\left(\omega-\frac{1}{2}\theta\right)F\left(\omega+\frac{1}{2}\theta\right)e^{jt\theta}d\theta \quad (1)$$

Here the kernel is 1. The autocorrelation with the lag variable, τ, produces the time-relative-time or temporal autocorrelation function given in equation (4). An important property of the Wigner distribution is that it is real with $W^*(t,\omega)=W(t,\omega)$. Also, the Wigner distribution gives a clear picture of the instantaneous frequency and group delay, which is not the case for the spectrogram. These are important for resynthesis [1,7].

2.1. The discrete pseudo-Wigner Distribution

The discrete implementation of the pseudo-Wigner distribution with a frequency smoothing window function $w(k)$, with length $M=2L-1$, $w(k)=0$ for $|k|\geq L$ is then defined by:

$$PWD\left(n,\frac{m\pi}{M}\right)=2\sum_{k=-L+1}^{L-1}g(n,k)p(k)e^{-2jk\frac{mn}{N}}, \quad m=0,\ldots,M \quad (2)$$

where

$$p(k)=w(k)w^*(-k) \quad (3)$$

and:

$$g(n,k)=f(n+k)f^*(n-k) \quad (4)$$

$g(n,k)$ is known as the temporal correlation function (TCF) or autocorrelation function. Equation (2) can be interpreted as the discrete Fourier transform of the autocorrelation function $g(n,k)$ with respect to *n* for each value of *m*.

2.1.1. Cross terms

Given a music signal model as follows:

$$f(t) = \sum_{k=1}^{M} A_k e^{j(w_k t + \phi_k)} \tag{5}$$

where k is the partial series index, t is time, and the k^{th} term in the summation represents a partial with constant amplitude A_k, frequency ω_k, and phase ϕ_k, the Wigner distribution is:

$$\begin{aligned} W_f(t,\omega) &= \sum_{k=1}^{M} A_k^2 \delta(\omega - \omega_k) \\ &+ \sum_{k=1}^{M} \sum_{l=k+1}^{M} A_k A_l \cos([\omega_k - \omega_l]t + \phi_k - \phi_l) \\ &\times \delta\left(\omega - \frac{(\omega_k + \omega_l)}{2}\right) \end{aligned} \tag{6}$$

The partials of $f(t)$ (*auto* terms) are given by the first term in equation (6). The second double summation indicates the cross terms, arising from products between partials, which lie between any pair of auto terms. The magnitude of the cross terms is the product $A_k A_l$ of the amplitudes of auto terms k and l and they oscillate at a frequency, $(\omega_k + \omega_l)/2$ equal to the difference between the frequencies of the two auto terms. For strictly harmonic signals, the cross terms form a partial series an octave below the fundamental, resulting in cross terms which fall at the same frequencies of and therefore corrupt the autoterms, and also cross terms at partial frequencies not in the original signal.

2.2. The Modal distribution

The modal distribution in equation (7) was designed to minimise these cross terms in equation (6) for music signals. The modal kernel consists of two different filter functions. The time-smoothing window, $h_{LP}(p)$, has the effect of smoothing the cross terms in the time direction, and the frequency-smoothing window, $g_{LP}(l)$, implements cross term suppression in cases of frequency modulation. $h_{LP}(p)$, is chosen to be a low pass filter with an upper cut-off just below the minimum frequency spacing in the distribution, this being the fundamental frequency for quasi-harmonic signals. The discrete form of the modal distribution is defined by:

$$M(n,k) = \sum_{n=-L+1}^{L-1} R_{f,l}(n,l) g_{LP}(l) e^{\frac{-j2\pi kl}{2L}} \tag{7}$$

where $R_{f,l}(n,l) = R_f(n-p,l) h_{LP}(p)$ is the time-smoothed temporal autocorrelation function (STCF). Computing the time-smoothing in the autocorrelation domain greatly reduces the number of DFTs that need to be performed. DFT's need to be computed only at *hop* steps that sample at a rate approximately equal to the period of the time smoothing window.

2.3. The Autocorrelation Function

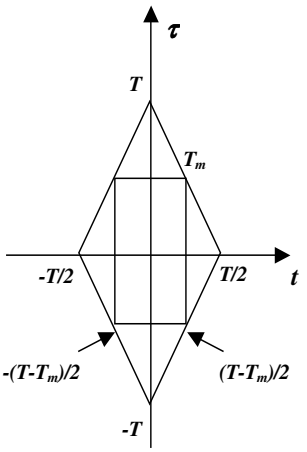


Figure 1: *Extent of the windowed autocorrelation function*

The autocorrelation function $g(n,k)$, represented by the diamond-shaped function in Figure 1, is sampled in time (t) at twice the Nyquist rate, or $2f_s$, and in relative-time (τ) at rate f_s. This function, then, has duration $2T$ in τ as shown in Figure 1. This requires that the discrete frequency index *k* in equation (2) be interpreted relative to this 2:1 sub-sampling rate [3]. With application of a 2-D kernel function, $2T_M$ represents the length of the frequency smoothing filter and the diamond-shaped region in the (t,τ) plane in Figure 1 is limited to the rectangular region [6]:

$$|t| > \frac{(T - T_m)}{2}, |\tau| > T_m \tag{8}$$

3. MODAL DISTRIBUTION SYNTHESIS METHOD

From equation (2) the inverse discrete transform is given by:

$$y(n,2k) = g(n,k)p(k) \tag{9}$$

resulting in the autocorrelation function written as:

$$f(n+k)f^*(n-k) = \frac{y(n,2k)}{2g(k)g^*(-k)} \tag{10}$$

or with appropriate change of variable as:

$$\begin{aligned} &f(n)f^*(m) = c(n,m) \\ &= y(\frac{n+m}{2}, n-m) \Big/ 2g\left(\frac{n-m}{2}\right).g^*\left(\frac{m-n}{2}\right) \end{aligned} \tag{11}$$

This outer-product formulation represents the product of two one-dimensional functions separable in *n* and *m* into the odd-or even-indexed sequences of signal samples. For the even-indexed samples, the outer-product formulation C_e can be written in matrix form as:

$$C_e = \begin{bmatrix} C_{0,0} \cdots C_{0,P} & C_{0,P+1} \cdots C_{0,L} \\ C_{P,0} \cdots C_{P,P} & C_{P,P+1} \cdots C_{P,L} \\ C_{P+1,0} \cdots C_{P+1,P} & C_{P+1,P+1} \cdots C_{P+1,L} \\ C_{L,0} \cdots C_{L,P} & C_{L,P+1} \cdots C_{L,L} \end{bmatrix} \quad (12)$$

where there are *P* known even samples and *L-P* samples to be recovered. This outer-product (OP) formulation is used in [1] for synthesis from overlapping blocks of C_e.

For signal synthesis from the modal distribution, however, only *hop* number of frames of the autocorrelation function are available and therefore the OP method cannot be directly applied. We derive an alternative method, which requires *hop* number of known samples to recover the sequence of odd- or even-indexed samples on a frame-by-frame basis.

3.1. Sub sampled autocorrelation function method

Synthesis of the signal samples from a sub sampled version of C_e is implemented in two stages: in the first stage, C_{e1}, processes all autocorrelation frames up to h_fft, half the DFT length (or $l = h\_fft/hop$ frames), where the size of each frame grows by hop number of samples and hop number of samples can be recovered from each frame. In the second stage, C_{e2} recovers $hop/2$ samples from all remaining frames. There are three cases only which must be processed separately:

i. the first frame of C_e contains the product $f_e(0)f_e^*(0)$ and so no processing is necessary

ii. the number of samples recovered from the second frame is *h/2*

iii. the number of samples recovered for frame *l+1* is:

$$h - a/2 \text{ where } a = (l+2) * h - (h\_fft - 1) \quad (13)$$

The matrix C_e can now be reformulated to take the hop step into account. For the even-indexed signal samples f_e we define the autocorrelation samples:

$$f_e(n)f_e^*(m) = c_e(n,m) = y\left(\frac{n+m}{2}, n-m\right) \Big/ 2g\left(\frac{n-m}{2}\right).g\left(\frac{m-n}{2}\right) \quad (14)$$

where $n = 0, hop, 2*hop, 3*hop, ..., t*hop$, and *t* is the total number of frames. Now we can write:

$$C_{e1} = \begin{bmatrix} C_{i_1,i_1}, & C_{i_1,i_1-2}, & C_{i_1,i_1-4} \cdots & C_{i_1,i_1-2hop-2} \\ C_{i_2,i_2}, & C_{i_2,i_2-2}, & C_{i_2,i_2-4} \cdots & C_{i_2,i_2-2hop-2} \\ C_{i_3,i_3}, & C_{i_3,i_3-2}, & C_{i_3,i_3-4} \cdots & C_{i_3,i_3-2hop-2} \\ \cdots & & & \\ C_{i_k,i_k}, & C_{i_k,i_k-2}, & C_{i_k,i_k-4} \cdots & C_{i_k,i_k-2hop-2} \end{bmatrix} \quad (15)$$

where $\langle i_1, i_2, \cdots, i_k \rangle = \langle 2*hop, 3*hop, \cdots, l*hop \rangle$, gives the hop frame index. Given a matrix $A = \lfloor a_0 a_1 \cdots a_p \rfloor$ of *p=hop-1* known even samples, and $X_e = diag(A)$, a diagonal matrix generated from *A*, all even indexed samples from C_{e1} can be determined by:

$$F_{e1} = C_{e1} / X_e \quad (16)$$

Next we can write:

$$C_{e2} = \begin{bmatrix} C_{j_1,j_1+\alpha_1}, & C_{j_1,j_1+\alpha_1-2}, & \cdots C_{j_1,j_1+\alpha_1-hop+2} \\ C_{j_2,j_2+\alpha_2}, & C_{j_2,j_2+\alpha_2-2}, & \cdots C_{j_2,j_2+\alpha_2-hop+2} \\ C_{j_3,j_3+\alpha_1}, & C_{j_3,j_3+\alpha_1-2}, & \cdots C_{j_3,j_3+\alpha_1-hop+2} \\ \cdots & & \\ C_{j_s,j_s+\alpha_v}, & C_{j_{s1},j_s+\alpha_v-2}, & \cdots C_{j_s,j_s+\alpha_v-hop+2} \end{bmatrix} \quad (17)$$

where $\alpha_1 = hop/2 - 1$, $\alpha_1 = \alpha_2 = \alpha_v$ and $\langle j_1, j_2, \cdots, j_s \rangle = \langle k+2, k+3, \cdots, t \rangle$ for even *hop,* and $\alpha_1 = hop/2 - 2$, $\alpha_2 = hop/2 - 1$ and $\alpha_v = \alpha_1$ or $\alpha_v = \alpha_2$ depending on *s*, for odd *hop*. The sequence $(j_i + \alpha_{1or2}, \cdots, j_i + \alpha_{1or2} - hop + 2)$ where $i = 1,2,\cdots s$, represents the *hop/2* known samples used to recover *hop/2* even-indexed samples from each row of C_{e2}. An identical formulation applies to the odd-indexed samples f_o and so need not be outlined.

4. RESULTS

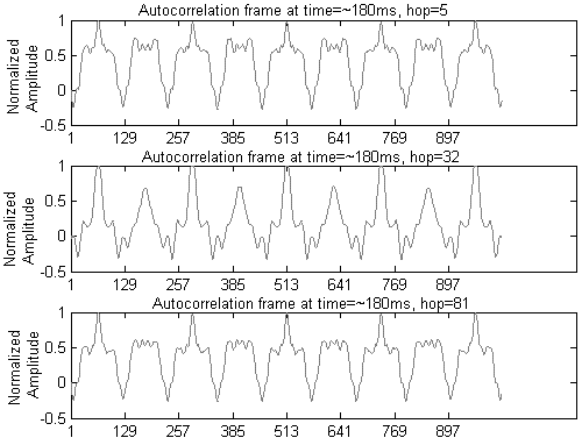


Figure 2: Autocorrelation function (for Bb Clarinet note G3) slices at time=180ms, for hop steps of 5, 32 and 81

Figure 2 shows TCF function relative-time slices at time 180ms, for a Bb clarinet G3 note of length 8000 samples with $f_s = 44100$ and $2T = 1024$ for hop sizes of 5, 32 and 81 respectively. In each case the peaks in the relative-time direction (horizontal axis) indicate the signal harmonics. For example, the

fundamental frequency can be seen from the 9 signal cycles in each plot, indicating a frequency of approximately $2f_s$ =9.1/1024 or ~196Hz (G3).

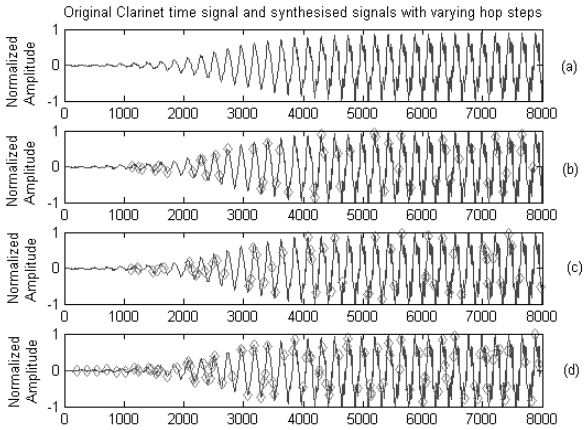


Figure 3: Comparison of (a) original Clarinet G3 note with synthesized Clarinet G3 note samples (b), (c) and (d) from respective autocorrelation functions in Figure 2. The diamonds in (b), (c), and (d) indicate gaps in the signal where samples could not be recovered.

Figure 3 shows the original signal (a), and the three signals recovered from the autocorrelation functions with hop steps of (b) 5, (c) 32, and (d) 81 respectively. The diamonds on plots (b)-(d) in Figure 3 indicate where zeros occur in the recovered signals. These missing samples are subsequently interpolated to avoid 'clicks' in the recovered signals. Hop sizes of arbitrary length were tested and in each case the synthesised samples recovered were identical to the original signal, apart from where zeros occurred, and the recovered signal was audibly indistinguishable from the sound of the original signal.

5. CONCLUSIONS AND FUTURE WORK

This frame-by-frame resynthesis method for Modal distributions exactly recovers the even- and –odd indexed signal samples for arbitrary hop steps. It provides an alternative signal recovery method for the Modal distribution based on the outer-product method in [1]. Current work focuses on a comparison of the outer-product approximation (OPA) method in [1] implemented for the Modal distribution using eigenvalue-eigenvector decomposition for signal recovery, with the method outlined in this paper. Immediate further work will implement signal filtering for the Modal distribution using these methods. Future work will also investigate the effect of the Modal distribution's smoothing kernel on this method and the possibility of signal modification in comparison with analysis-synthesis approaches. Finally, this frame-by-frame approach readily integrates into the SndObj library's Modal distribution routine [6,7], thus allowing a streaming implementation of Modal distribution synthesis in conjunction with many of the tools necessary for sound analysis and modification such as time stretching and vocoding.

6. REFERENCES

[1] K.-B., Yu,. and S. Cheng, " Signal synthesis from pseudo Wigner distribution and applications", *IEEE Trans. Acoust., Speech, Signal Processing,* Vol. ASSP-35, pp.1289-1302, Sept. 1987.

[2] W. Mecklenbraukner, and F. Hlawatsch (Editors), "*The Wigner Distribution – Theory and Applications in Signal Processing*", 1997, Elsevier Science B.V. pp. 135-209.

[3] W. J. Pielemeier, and G. Wakefield, "A high-resolution time-frequency representation for musical instrument signals", *Journal of Acoustical Society of America",* 99(4), Pt. 1, April 1996.

[4] G. F. Boudreaux-Bartels, "Time-Varying Filtering and Signal Estimation Using Wigner Distribution Synthesis Techniques", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP-34, No. 3, June 1986.

[5] L. Cohen, *Time Frequency Analysis*. Prentice-Hall, New Jersey, 1995.

[6] V. Lazzarini, "The sound object library". *Organised Sound 5 (1).* Pp. 35-49, 2000.

[7] T. Lysaght, V. Lazzarini, and J. Timoney, 'A Streaming Object Oriented Implementation of the Modal Distribution' *In Proc. Of International Computer Music Conference (ICMC-05)*, Barcelona, Spain, September 5-9, 2005.

FREQUENCY SLOPE ESTIMATION AND ITS APPLICATION FOR NON-STATIONARY SINUSOIDAL PARAMETER ESTIMATION

Axel Röbel

IRCAM-CNRS-STMS, Analysis-Synthesis Team
Paris, France
roebel@ircam.fr

ABSTRACT

In the following paper we investigate into the estimation of sinusoidal parameters for sinusoids with linear AM/FM modulation. It will be shown that for linear amplitude and frequency modulation only the frequency modulation creates additional estimation bias for the standard sinusoidal parameter estimator. Then an enhanced algorithm for frequency domain demodulation of spectral peaks is proposed that can be used to obtain an approximate maximum likelihood estimate of the frequency slope, and an estimate of the amplitude, phase and frequency parameter with significantly reduced bias. An experimental evaluation compares the new estimation scheme with previously existing methods. It shows that significant bias reduction is achieved for a large range of slopes and zero padding factors. A real world example demonstrates that the enhanced bias reduction algorithm can achieve a reduction of the residual energy of up to 9dB.

1. INTRODUCTION

Additive (or sinusoidal) models are often used for the representation, analysis or transformation of music or speech signals [1, 2]. An important step that is necessary to obtain the sinusoidal model consists of the estimation of the amplitude, frequency and phase of the sinusoids from the peaks of the discrete Fourier transform. The estimation is rather simple as long as the signal is stationary. A standard method for this estimation is the quadratically interpolated FFT (QIFFT) [3]. The QIFFT estimator uses the bin at the maximum of each spectral peak together with its two neighbors to establish a 2nd order polynomial model of the log amplitude and unwrapped phase of the peak. The amplitude and frequency estimates of the sinusoid that is related to the spectral peak are then derived from the height and frequency position of the maximum of the polynomial. The evaluation of the phase polynomial at the frequency position provides the estimate of the phase of the sinusoid.

For non-stationary sinusoids the parameter estimation becomes more difficult because the QIFFT algorithm is severely biased whenever the frequency is not constant. The term bias refers to the systematic estimation error. It describes the offset of the estimator that exists even if no measurement noise is present. For the partials in natural vibrato signals the estimation bias of the QIFFT estimator accounts for a significant amount of residual energy. It is the major reason for the perceived voiced energy in the residual of vibrato signals. A number of algorithms with low estimation bias for non stationary sinusoids have been proposed. Algorithms that try to implement a MLE are generally assuming that the amplitude of the sinusoids are constant. As example we refer to an algorithm that is based on signal demodulation employing an initial search over a grid of frequencies and frequency slopes and a final fine-tuning of the parameters using an iterative maximization of the amplitude of the demodulated signal [4]. Similar as for multi-component signals with stationary sinusoids the MLE of sinusoidal parameters for multi-component signals with FM modulated sinusoids is rather costly as in this case a highly nonlinear and high dimensional cost function needs to be maximized [5]. Due to the computational savings and despite the fact that windowing reduces the estimator efficiency the windowing technique is generally preferred if the signal contains more than a single sinusoid. The algorithms that employ analysis windows for the parameter analysis of AM/FM modulated sinusoids generally rely on the fact that the analysis window is approximately Gaussian such that a mathematical investigation becomes tractable [6, 7, 3]. The method presented in [3] is special in that it tries to extend its range to other analysis windows by means of a set of linear bias correction functions. The resulting estimator is computationally rather efficient and achieves small bias for standard windows as long as the zero padding factor is sufficiently large (≥ 3) and the frequency chirp rate is relatively small.

In the following paper we present a bias correction scheme for sinusoidal parameter estimation of sinusoids with linear AM/FM modulation. As a first step we provide a mathematical foundation for the conjecture that linear amplitude modulation does not create any additional bias for the QIFFT estimator. With respect to bias reduction we may therefore ignore the amplitude modulation of the signal. Then we extend an initial version of our bias reduction method that has been proposed originally in [10]. The basic ideas of the algorithm are similar to [4] in that the algorithm is based on signal demodulation and maximization of the amplitude of the demodulated signal to find the sinusoidal parameters. In contrast to [4] however, the algorithm allows the use of a analysis window and does not use time domain demodulation. Therefore, it can be applied if the signal contains more than a single sinusoid. Moreover, the initial 2-dimensional grid search of the algorithm presented in [4] is avoided due to the fact that first, a simple and efficient initial estimate of the frequency slope estimate is used, and second, the frequency and frequency slope estimation have been decoupled. After demodulating the frequency slope the standard QIFFT estimator can be applied to obtain an estimate of the sinusoidal parameters. Due to the fact that the QIFFT estimator has small bias for constant frequency sinusoids the resulting estimate is significantly improved. In [10] it has been shown that demodulation can be achieved by means of spectral deconvolution using only the peak to be analyzed and a properly selected and scaled demodulation kernel. In the original version the frequency slope estimate was entirely handled by the frequency slope estimator in [3].

The version to be presented here is a refined version of the original demodulation algorithm. The enhancements include a

new procedure to improve the initial estimate of the frequency slope reducing the remaining bias for large frequency slopes. Furthermore, the constraint to use the same analysis window for the signal spectrum and the demodulation kernel has been removed. Accordingly, it becomes possible to trade-off bias against noise sensitivity. A computationally efficient version of the algorithm using precomputed and linearly interpolated demodulation kernels is presented. We describe an experimental comparison of the new frequency slope estimator with the previous version and the approach presented recently in [3] and an experimental evaluation of different bias reduction schemes for a real world vibrato signal.

The organization of the article is as follows. In section **2** we will show how the bias of the standard estimators is related to the frequency slope. In section **3** we will describe the demodulation scheme and the improved frequency slope estimator. In section **4** we present experimental results for the frequency slope estimation algorithm as well as for the bias reduction scheme by means of comparing the results of different algorithms. Furthermore we compare different bias reduction methods by means of comparing the residual energy of the sinusoidal model of a real world vibrato signal. In section **5** we conclude with an outlook on further improvements.

2. ESTIMATION BIAS

The signal model that will be used in the following assumes a linear evolution for amplitude and frequency trajectories. Accordingly, a complex discrete time sinusoid can be represented as

$$s(n) = (A + an)\exp(i(\phi + 2\pi\omega_0 n + \pi D n^2)). \qquad (1)$$

Here A is the mean amplitude of the signal and a is the amplitude slope. ϕ is the phase of the sinusoid at time $n = 0$, ω_0 is its mean frequency and D is the frequency slope. Note, that all frequency values are normalized with respect to the samplerate. The center of the analysis window is is located at time 0 such that an ideal estimator should provide (A, ω_0, ϕ) as estimates for amplitude, frequency, and phase. The model equation (1) is necessarily time limited due to the fact that we assume $A + an > 0$ for all sample positions n that are used in a signal analysis.

As introduction into the problem we will summarize the sources of bias that are known to exist for the standard QIFFT estimator and discuss there implications in the context of parameter estimation for linear AM/FM modulated sinusoids.

First, there is the use of a second order model for interpolating the spectral bins. While this is systematically wrong for the present sinusoidal model, it does not have any direct relation to the fact that the sinusoidal parameters are varying. Because the QIFFT algorithm will be used extensively, it is nevertheless important to reduce this type of bias as far as possible. This can be achieved by means of zero padding the analysis window or, as demonstrated recently, by means of simple bias correction functions [8].

Second, there is the cross component bias that is due to other sinusoidal components. The technique that is generally used to reduce this bias is windowing. The analysis window reduces the sidelobes of the sinusoidal components such that the cross component bias of distant sinusoidal components can be effectively reduced. Note however, that the reduction of the sidelobe amplitudes is always accompanied by an increased mainlobe width. Therefore, the windowing technique will slightly increase the cross component bias for nearby components. Moreover, due to the tapering of the signal at the frame borders the noise sensitivity of the parameter estimation is slightly increased. In the following we will assume that the sinusoidal components are resolved such that the frequency distance between two sinusoids is always larger than the width of the mainlobe of both components. In this case the cross component bias will stay nearly the same for stationary and non-stationary components such that the cross component bias will only change marginally with the modulation of the sinusoids.

Third, there is the bias due to the non-stationary parameters. For the sinusoidal model in equation (1) and a Gaussian analysis window the bias has been analyzed mathematically in [7]. The result shows, that the QIFFT algorithm suffers from additional bias due to parameter variation only if the frequency slope $D \neq 0$. In this case, the estimation of all three basic parameters are biased and the bias increases with the absolute value of D.

To study the dependency of the estimation bias on the frequency slope for arbitrary analysis windows we split the sinusoidal model in equation (1) into two parts, a sinusoid with constant amplitude A and sinusoid with mean amplitude 0 and amplitude slope a. Then we investigate the properties of the spectra of the individual parts and use the linearity of the Fourier transform to draw conclusions for the complete spectrum. We first write the DFT of the signal equation (1) using analysis window $W(n)$ as follows

$$S(w) = \sum_{n=-\infty}^{\infty} W(n)(A+an)e^{i(\phi+2\pi\omega_0 n+\pi D n^2)}e^{-i2\pi\omega n}. \qquad (2)$$

Assuming the analysis window to be even symmetric we can make use of the symmetry relations and remove all parts of the sum in equation (2) that are odd symmetric in n. As a result equation (2) simplifies into

$$S(\omega) = S_c(\omega) + S_l(\omega) \quad \text{with} \qquad (3)$$

$$S_c(\omega) = Ae^{i\phi} \sum_{n=-\infty}^{\infty} (W(n)\cos(2\pi(\omega_0 - \omega)n)e^{i\pi D n^2} \qquad (4)$$

$$S_l(\omega) = ae^{i\phi} \sum_{n=-\infty}^{\infty} W(n)ni\sin(2\pi(\omega_0 - \omega)n)e^{i\pi D n^2} \qquad (5)$$

Here S_c represents the spectrum of the constant amplitude part and S_l represents the spectrum of the linear amplitude part of the sinusoid.

For the discussion of equations (3-5) we assume the coordinate system of the amplitude and phase spectra to be shifted using the translation $\omega' = \omega - \omega_0$. Accordingly, the frequency origin of ω' is located at the sinusoidal frequency ω_0. For $D = 0$ the amplitude of the spectra of both parts will be even functions with the spectrum of the second part being 0 at the origin. $S_c(\omega')$ and $S_l(\omega')$ have a local maximum respectively minimum at the origin. The phase of $S_c(\omega')$ is constant with value ϕ within the mainlobe. The phase of $S_l(\omega')$ is odd, it consists of two constant parts (with value $\phi \pm \pi/2$) with a phase jump of π right at the origin. The sum of $S_c(\omega')$ and $S_l(\omega')$ has even amplitude and odd phase ϕ with the value $Ae^i\phi$ at the origin. Depending on the ratio of A and a the spectrum may present either a local maximum or minimum at the origin. For all common analysis windows and the sinusoidal model in equation (1) the resulting spectrum has a maximum. As our first result we may conclude that for $D = 0$ the QIFFT estimator provides results that are biased only by the first two sources of bias mentioned above and that the time varying amplitude $a \neq 0$ does not add any additional bias.

For $D \neq 0$ the factor $e^{i\pi Dn^2}$ adds an even phase to the elements of the sum. As a result the magnitude of $S_c(\omega')$ and $S_l(\omega')$ does keep all the characteristics discussed above, notably even symmetry and extreme value characteristics (maximum and minimum). The (unwrapped) phase spectra however are no longer (locally) constant, but both phase spectra have an additional even phase function superimposed. The phase offset of $S_c(\omega')$ does not vanish at the origin and by consequence the phase is biased already for $a = 0$. For $a \neq 0$ the even symmetric phase offset that is applied to $S_l(\omega')$ will destroy the even symmetry of the magnitude of $S(\omega')$ such that the peak maximum moves away from the origin, and therefore, the amplitude and frequency estimates of the QIFFT estimator are no longer correct. Accordingly, the QIFFT estimator suffers from additional bias quite similar as has been shown for the Gaussian window in [7].

3. REDUCING THE BIAS

In the previous section we saw that the source of the bias of the QIFFT estimator is the frequency slope of the sinusoid. A conceptually simple approach to estimate the parameters (A, ϕ, ω) of a sinusoid related to a spectral peak requires two steps:

1. estimate the frequency slope,
2. demodulate the sinusoid and use the QIFFT estimator to find the sinusoidal parameters.

Note, that this approach is in principle equivalent to the MLE for constant amplitude linear FM signals described in [4]. Because the demodulation technique is used for the frequency slope estimation we will first discuss the frequency domain demodulation algorithm. In the following section the frequency slope estimation is described.

3.1. Demodulation

The main objective of the present algorithm is to provide a means to demodulate the sinusoid using only the part of the spectral peak that is accessible for analysis. Because the sinusoidal peak is covered by noise this part will generally be the part of the mainlobe exceeding the noise level. Initially, we assume we are given a frequency slope estimate $\hat{D} = D$ for a peak that is part of a signal spectrum.

In time domain the demodulation can be achieved simply by multiplication with a demodulator signal

$$y(n) = e^{-i\pi \hat{D} n^2}. \tag{6}$$

Multiplication of the signal in equation (6) with the signal equation (1) will remove the frequency slope and keep all other parameters unchanged such that the QIFFT algorithm can be applied. However, the signal we are interested in is observable only via the part of its mainlobe that constitutes the observed spectral peak.

The demodulation algorithm that uses the observed peak to demodulate the sinusoid will be described in the frequency domain using as sources the spectral peak to be analyzed and the spectrum of the deconvolution signal. Assume $S(k)$ is the N-point DFT of the sinusoid to be analyzed and $Y(k)$ the DFT of the demodulator signal. All DFT spectra are calculated such that the origin of the DFT basis functions is in the center of the analysis window. The signal analysis window is $w_s(n)$ and the demodulator signal is windowed using $w_y(n)$. To obtain the demodulated sinusoid spectrum $X(k)$ we would need to compute the circular convolution

$$X(k) = C\frac{S(k) \circledast Y(k)}{N}, \tag{7}$$

where C is a normalization factor taking into account windowing effects. The demodulator window w_d will be multiplied with the signal window such that the resulting spectrum contains as effective window the product window $w_y(n)w_s(n)$. Therefore, proper normalization would be achieved by means of setting $C = 1/\sum_n (w_y(n)w_s(n))$.

Due to the fact that only part of the sinusoid spectrum is available the normalization factor needs to be adapted. Assume the peak under investigation is denoted by $P(k)$. $P(k)$ is part of the spectrum $S(k)$ and covers B bins. To estimate the impact of the missing part we create a spectral model of the observed sinusoid assuming the initial slope estimate is correct

$$P_m(k) = \sum_n w_s(n) \exp(i\pi \hat{D} n^2) \exp(-\frac{2\pi j}{N} kn), \tag{8}$$

and select a subset $\bar{P}_m(k)$ of B bins around the center frequency $k = 0$. [1] The required normalization factor can now be approximately estimated as

$$C = \frac{1}{\max_k(|\bar{P}_m(k) \circledast Y(k))}. \tag{9}$$

Now we can replace $S(k)$ in equation (7) by $P(k)$ and demodulate using the corrected normalization factor C. Some remarks are in order:

- The correction factor will be more precise (lower bias) for demodulator windows that concentrate more energy in the B-bin wide band around frequency 0 of the spectrum. This calls for low side lobes. The demodulator window, however, will as well be applied to the signal. As a result the estimator sensitivity to noise will increase. Accordingly the demodulator window allows to trade-off noise sensitivity and bias. The experimental investigation suggests that the use of the Hanning window as demodulator window w_d is a favorable choice for all analysis windows w_s.
- The compensation of the normalization factor assumes that the amplitude slope $a = 0$ and that the peak model is cut symmetrically with respect to the peak center. To create an optimal correspondance between the compensation factor and the missing part of the signal it is preferable if the spectral peak $P(k)$ that is used for demodulation is as close as possible to the peak model that is used to derive the compensation factor. The comparison of a number of strategies that may be employed to extract the observed peak from the spectrum we found that cutting the peak such that its left and right magnitude have approximately the same value creates the smallest bias. Besides the fact that this method achieves perfect compensation for $a = 0$ there is a second advantage of this method that is related to the impact of the background noise. Assuming the background noise energy to be constant and understanding the maximum border amplitude of the peak as a very rough indicator of the background noise level we may conclude that cutting the peak at its maximum border level could be beneficial because it avoids the parts of the signal where the background noise is dominant.

[1] If B is even the resulting model is not symmetric!

- For parameter estimation from demodulated peaks with the QIFFT estimator it is essential to use the bias correction functions proposed in [8] with correction factors adapted to the effective window $w_y(n)w_s(n)$.

Our experimental investigation shows, that the demodulation kernels $Y(k)$ can be precalculated for a fixed grid of frequency slopes and then linearly interpolated to obtain an approximate demodulation kernel for any given slope. If the length of the analysis windows is M a frequency slope grid with step size $0.025/M^2$ is sufficient to produce estimates that are nearly indistinguishable from the results produced with the non interpolated kernels. To use the complete information that is available in the observed peak we use deconvolution kernels of length $2B + 1$ centered around the maximum of the deconvolution spectrum.

The implementation of the deconvolution can be done in the frequency domain (as described) or in the time domain. Time domain implementation would probably be more efficient if at least the demodulation kernel could be directly stored in the time domain. The possibilities of time domain interpolation of the demodulation kernels have not yet been studied, we believe however, that time domain interpolation would require on the fly generation of the complex kernels from interpolated phase functions. Due to the linearly modulated frequency of the demodulation kernels this will most likely be less efficient than the frequency domain implementation that has been described above.

3.2. Frequency slope estimation

As mentioned above the maximum likelihood (ML) frequency slope estimator for constant amplitude linear FM sinusoids maximizes the amplitude of a demodulated peak [4]. Accordingly the maximization of the amplitude of the demodulated peak using the demodulation algorithm described above can be considered an approximate MLE as long as the amplitude slope is sufficiently small.

To avoid the search of a large grid of frequency slopes we propose to use an approximate initial estimate of the frequency slope $\hat{D}$, and then to use the frequency slope estimate and two slopes with $\hat{D} \pm D_o$ to create three different demodulations of the observed peak. From the amplitudes of these demodulated peaks a 2nd order polynomial model of the relation between frequency slope and demodulated amplitude can be derived. The maximum of this polynomial is expected to provide a refined estimate of the frequency slope.

The open question we need to address is: how do we get an approximate estimate of the frequency slope? Given the highest order coefficients α_ϕ and α_A of the QIFFT polynomial for amplitude (A) and phase (ϕ) of the peak under investigation the frequency slope estimate for a Gaussian analysis window is [3, 9]

$$\hat{D} = \frac{\alpha_\phi}{\alpha_\phi^2 + \alpha_A^2}. \tag{10}$$

Note the remarkable fact, that the same estimator has been obtained for exponential amplitude evolution in [3] and for a first order approximation of the spectrum of a sinusoid with linear amplitude evolution in [9]. The fact that the amplitude evolution function does not affect the frequency slope estimator leads us to suppose that that equation (10) will provide useful estimates for other windows than the Gaussian window as well. The argument here is that the signal that is obtained after the analysis window has been applied can always be considered to be equivalently generated by means of a Gaussian analysis window and a sinusoid with appropriately modified amplitude evolution. Because the desired frequency estimate does not change with the amplitude evolution of the sinusoid and because the estimator equation (10) appears to be rather insensitive to small changes of the amplitude evolution of the sinusoid it will be considered as approximate estimator for the frequency slope for arbitrary analysis windows.

The free parameter to select is the frequency slope offset D_o. In general a polynomial approximation improves when the approximation range is decreased. This would call for a small D_o. In the present case, however, the relation between demodulation slope and amplitude of the demodulated peak is covered by measurement noise (due to estimation errors of the amplitude of the demodulated peak, due to the partially observed sinusoidal spectrum, and due to the sampling of the Fourier spectrum by the DFT). The final selection of the D_o parameter will be discussed in section **4.1**.

The precision of the frequency slope estimate that is obtained from the maximum of the polynomial is slightly, but consistently improved if the polynomial model is not constructed for the demodulated amplitudes $\hat{A}_i$ but for $\hat{A}_i\sqrt{C_i}$ where C_i is the normalization factor from equation (9). Up to now a theoretical explanation of this experimental finding has not yet been found. Using $\sqrt{C}$ to calculate the demodulated amplitudes will obviously create biased amplitude estimates. For the problem of slope estimation it appears to improve the fit of the polynomial model and therefore, it will be preferred. After the slope has been determined from the maximum of the polynomial a re-normalization can be performed if the amplitude of the supporting points is required.

4. EXPERIMENTAL EVALUATION

The proposed parameter estimation procedure will be evaluated by means of comparing it to the bias correction algorithm proposed in [3] for which Gaussian and Hanning analysis windows are used. The results of that algorithm are denoted as *AS*. Furthermore we use the original version of the demodulation estimator according to [10]. (denoted as *DE*) and the new version that includes the slope enhancement and uses the Hanning window for all demodulation kernels (denoted as *DS*).

The window type that is used will be indicated by adding the letter *G* for Gaussian or *H* for Hanning or *X* for both to the estimator shortcut. The window applied to the demodulation kernels will be equal to the analysis window for *DEX* and Hanning for *DSX*. The Gaussian analysis window is cut such that it has a length of 8σ with σ being the standard deviation of the Gaussian. To facilitate orientation we display the results of the QIFFT estimator as well as the Cramer-Rao bounds for second order polynomial phase estimation with that have been presented in [11]. Note, however, that these bounds have been found for constant amplitude polynomial phase signals, such that they can only be used to provide an approximate idea of the estimator efficiency.

In the experiments we use synthetic test signals with a single sinusoid according to equation (1) with $A = 1$, ω_0 randomly sampled from a uniform distribution over the frequency range $[0.2, 0.3]$, ϕ randomly chosen from a uniform distribution between $[-\pi, \pi]$, and varying slopes a and D. The analysis window covers $M = 1001$ samples in all cases. The frequency slope D is selected from a uniform distribution over interval $[-D_{max}/M^2, D_{max}/M^2]$. Similarly the amplitude slope a is sampled from a uniform distribution over the range $[-a_{max}/M, a_{max}/M]$. The slope ranges are considered realistic for real world signals. Note, that in harmonic signals the frequency slope scales with the partial number

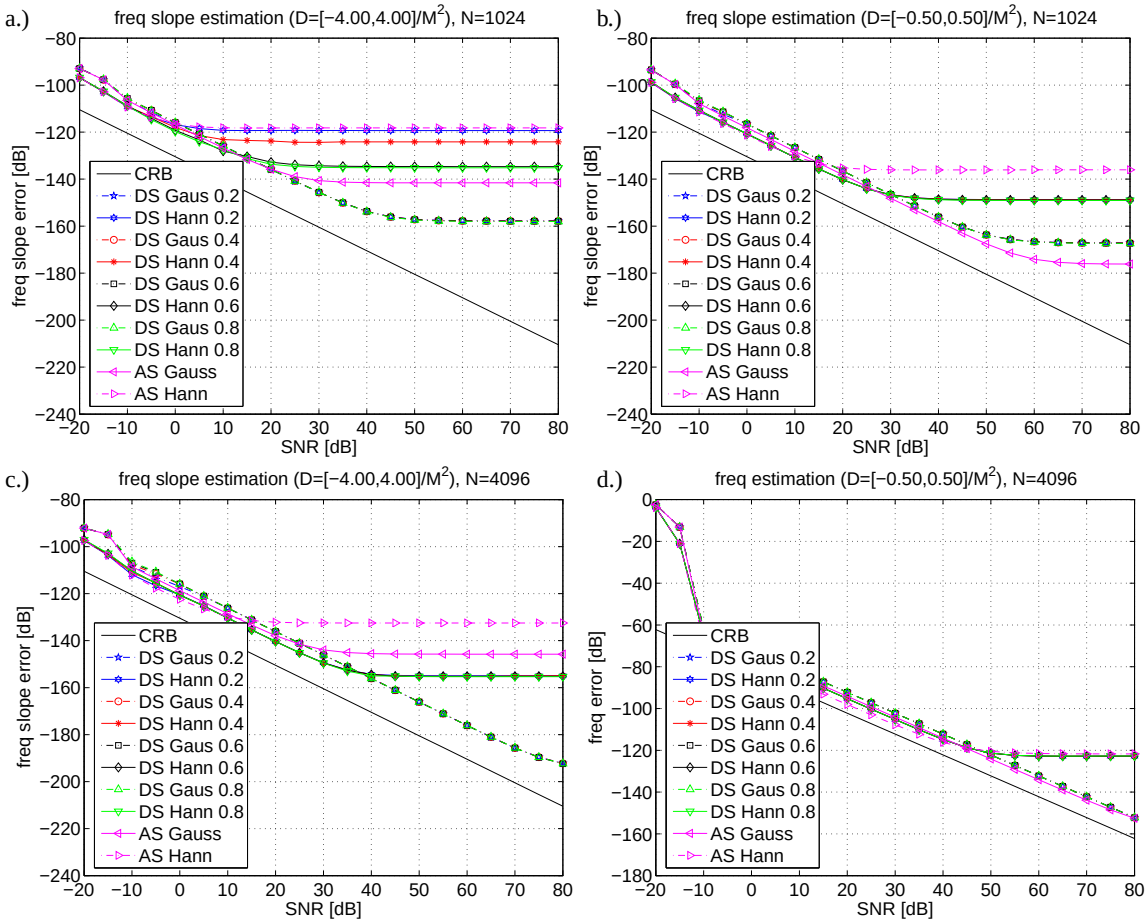


Figure 1: *Comparison of the frequency slope estimation errors for the* DSX *estimator with varying slope offset* D_o *and the* ASX *estimator. Window size is* $M = 1001$ *and sinusoids with strong (a,c) and weak (b,d) amplitude and frequency modulation are considered. DFT size is* $N = 4096$ *(a,b), and* $N = 1024$ *(c,d). The CRB for constant amplitude polynomial phase signals is displayed as lower limit. Algorithms using a Gaussian/Hanning window are distinguished by means of solid/dashed lines. See text form more details.*

.

such that for high partials extreme slopes may arise.

Note, that the implementation of the algorithm used for the experimental investigation uses linearly interpolated demodulation kernels as proposed in section 3.1.

4.1. Frequency slope estimation

In the first experiment we investigate into the frequency slope estimation. In Figure 1 we compare the enhanced demodulator *DSX* with the *ASX* method according to equation (10). Because the *DEX* estimator uses the frequency slope estimate provided by *ASX* directly we don't consider *DEX* here. We use two different zero padding factors (FFT size $N = 1024$ and $N = 4096$) and two different sets of modulation ranges, the strong modulation is using $D_{max} = 4$ and $a_{max} = 1$, while for weak modulation we select $D_{max} = 0.5$ and $a_{max} = 0.15$. Note, that the weak modulation range approximately covers the interval for that the *ASH* bias correction has been derived in [3]. The *DSX* estimator is operated with a set of demodulation offsets $D_o \in [0.2, 0.4, 0.6, 0.8]/M^2$.

The results of the experiment are shown in Figure 1. There are a number of conclusions that can be drawn from these figures. First, we find that for strong modulation the *DSX* method has significantly lower bias then the *ASX* method. Second, we observe that for the Hanning window the *DSH* estimator achieves a reduction of the estimation bias by $2 - 30$dB. The smallest improvement is achieved for weak modulation and large oversampling factor. The only case where the *ASX* estimator significantly outperforms *DSX* is weak modulation with small oversampling factor and Gaussian analysis window. This could have been expected because the *ASG* estimator is exact for the Gaussian analysis window and the small oversampling factor does not influence this estimator. As expected the Hanning window has larger bias than the Gaussian window but at the same time it is less sensitive to noise by about 4dB. In general the *DSX* are more sensitive to noise by about $2 - 3$db.

Considering the demodulation offset D_o we find that the offset has a significant impact only for strong modulation with small oversampling factor and Hanning window. This is related to the

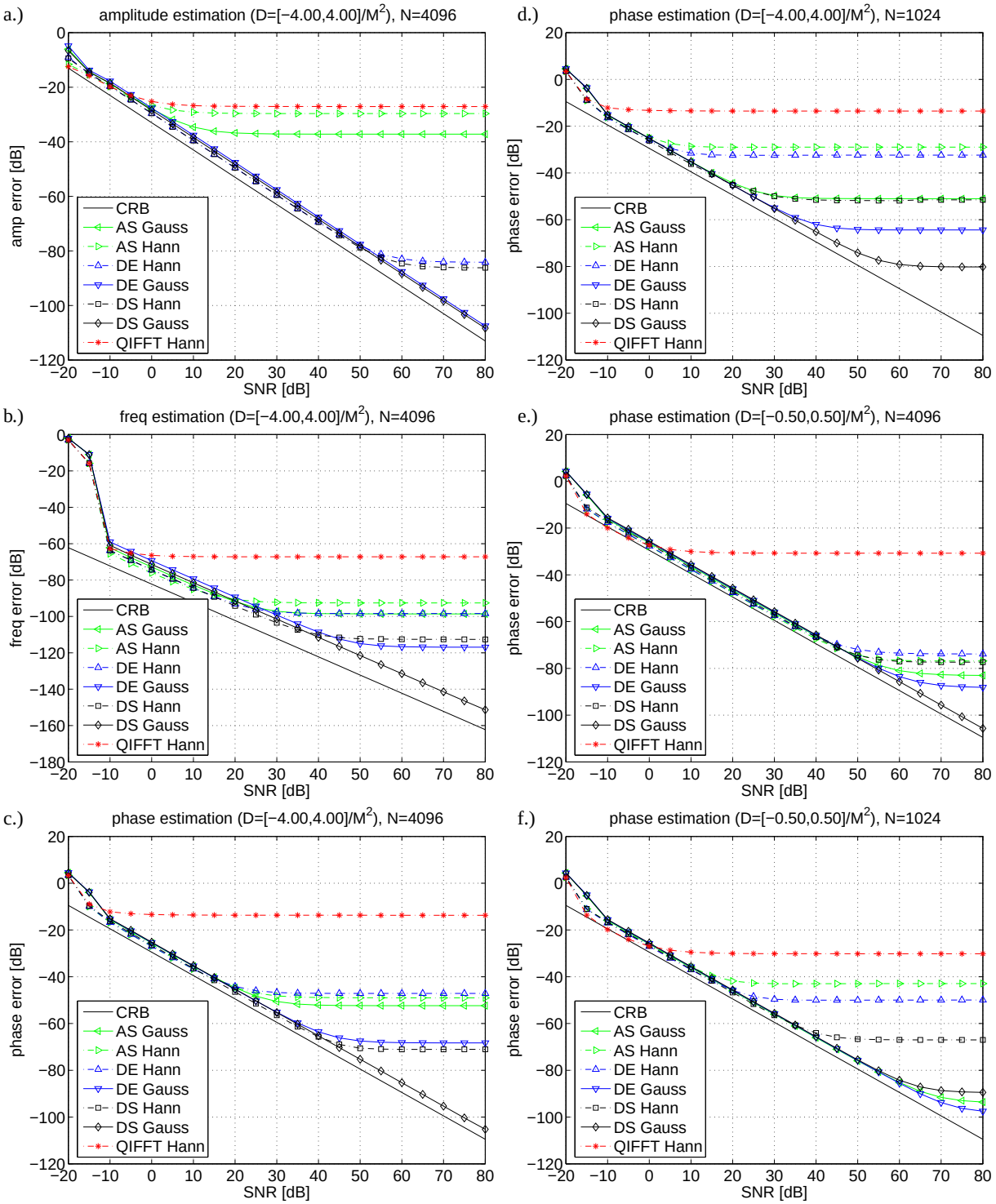


Figure 2: *Comparison of the estimation errors for the different parameter estimators using window size* $M = 1001$ *and FFT size* $N = 4096$ *and (strong) linear AM/FM with* $D_{max} = 4$ *and* $a_{max} = 1$ *(a-c). Figures (d-f) show phase estimation errors for different modulation limits and FFT sizes. The CRB for constant amplitude polynomial phase signals is displayed as lower limit. Algorithms using a Gaussian/Hanning window are distinguished by means of solid/dashed lines. See text form more details.*

.

fact that the initial frequency slope estimate of the *ASH* that is the basis of the slope refinement in *DSX* is rather bad, such that the model needs to compensate a larger range of slope errors. Moreover the amplitude estimation is less precise for smaller oversampling factors such that a larger slope offset may be required to obtain a polynomial model that captures the underlying relations. For $D_o = 0.5$ we get nearly optimal results for all cases which is why we select this value for the following experiments.

4.2. Bias correction

After having discussed the properties of the frequency slope estimation we now investigate into the main topic of this paper, the bias reduction. Due to space constraints we will only discuss a few of the experiments we have conducted. We will discuss the results for all parameters for strong modulation with $D_{max} = 4/M^2$ and $a_{max} = 1/M$ and an FFT size of $N = 4096$. Furthermore we select the phase bias reduction as an example and discuss the bias reduction for the phase estimate for weak and strong modulation and FFT sizes $N = 1024$ and $N = 4096$.

The results of the bias reduction for strong modulation and $N = 4096$ are displayed in the left column of Figure 2. As expected the amplitude estimate a.) of ASX is strongly biased due to the fact that the amplitude trajectory model does not match the signal. DEX and DSX are both similar and better then ASX. Note, that the improved frequency slope estimate of DSX hardly improves the amplitude estimate compared to DEX and that the increase of the noise sensitivity of DEX and DSX is negligible. For frequency b.) and phase estimation c.) DSX has by far the smallest bias (compared to the other estimators using the same analysis window). DEH and ASH perform approximately similar for both for frequency and phase estimation. Given that DEX and ASX estimators both use the same frequency slope estimate this shows that the bias of these two estimators is due to the error in the frequency slope estimate which is improved by the refined slope estimate of DSX.
The increase of the noise sensitivity for the demodulation algorithms is negligible for phase estimation. For the frequency estimator the use of the Hanning window instead of the analysis window is clearly diminishing the noise sensitivity when the analysis window is Gaussian.

The right column of Figure 2 shows the phase bias removal for all the experimental settings that were used in the evaluation of the frequency slope estimation. A close inspection of the results reveals that the performance of the bias removal is directly related to the performance of the frequency slope estimation. This can be expected because any error in the frequency slope estimate will translate into an error in the bias correction algorithm.

As a summary of the experimental investigation of the algorithm using synthetic signals we conclude that compared to the QIFFT estimator all the bias reduction algorithms dramatically reduce the estimation bias. Compared to the recent *ASX* estimator the simple and enhanced demodulation algorithm both provide a significant reduction of the estimation bias especially if the range of the modulation is not confined to the rather limited range of values that has been considered in [3]. Comparing the *DEX* and *DSX* algorithms we have shown that the enhanced slope estimation has a direct and significant impact on the bias of the sinusoidal parameters. Due to the fact that the frequency slope bias of the *DEX* algorithm increases with the modulation we expect that the *DSX*

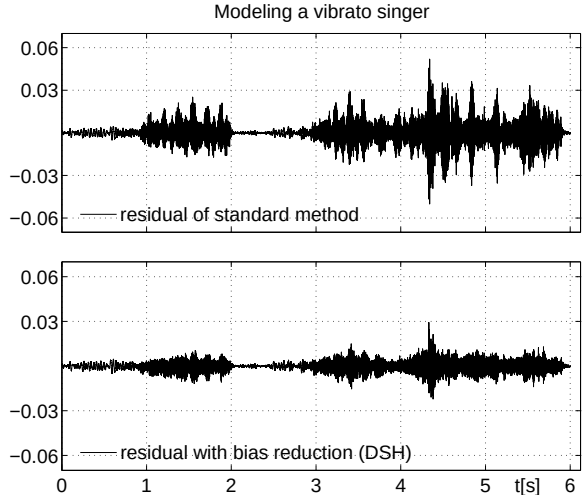


Figure 3: *Residual signal of a vibrato tenor singer using QIFFT estimator (top) and the enhanced demodulation method DSH (bottom).*

estimator is especially advantageous if the modulation is strong. The possibility to freely select the demodulator window improves the noise sensitivity in case the Gaussian window is used as analysis window.

4.3. A real world example

To demonstrate that the advantages of the proposed estimator are effective in real world situations we have implemented the bias reduction methods in a complete additive modeling system. The theoretical investigation has been restricted to cover the case of resolved sinusoids, only. For real world applications, however, the algorithm has to prove that it will act gracefully when the underlying model no longer holds (transients, unresolved sinusoids due to reverberation, ...). The major problem in real world signals is related to the fact that the enhanced frequency slope estimation described in 3.2 may produce extreme values whenever the underlying signal model does not match the observed peak. In these cases the method may for example try to model the transient or nearby sinusoids by means of extreme slopes.
To prevent the degeneration of the estimator we use a number of tests that are designed to allow us to detect the cases for that the signal model that is used to analyze the peak does not hold. The tests that verify the reliability of the second order polynomial model of the relation between demodulation slope and amplitude are: verification that the extremum of the polynomial model is a local maximum, verification that the amplitude that is obtained with the optimal demodulation slope is larger than the amplitude obtained with the initial slope estimate, verification of that the slope offset to reach the optimal slope is within $\pm 2D_o$. If one of these tests fails the polynomial representation of the slope and amplitude relation is considered unreliable and the *DEX* estimator is used as a fallback.

The test that verifies the validity of the linear AM/FM sinusoidal representation is based on the center of gravity of the energy (the mean time) of the signal related to the spectral peak under investigation. If the mean time is larger then the maximum mean time that can be expected for the signal model equation (1) then

we can assume that the peak is related to a sinusoid with transient amplitude evolution [12]. In this situation the exponential amplitude evolution used by the *ASX* estimator is more appropriate than the linear AM and therefore the *ASX* estimator is used. Note, that the *ASX* and *DEX* estimators are sub modules that are required for the *DSX* estimator anyway such that the fallback solutions do not require additional costs in terms of implementation or calculations.

freq band	ASH	DEH	DSH
full	-4.19	-4.72	-5.04
0-2kHz	-3.13	-3.75	-4.05
2-4kHz	-7.32	-8.40	-9.33
4-6kHz	-5.78	-6.90	-7.32

Table 1: *The reduction of the energy of the residual signal obtained with the different bias reduction algorithms. The performance of the algorithms varies with the frequency band.*

For the last experiment we compare the estimators by means of the energy of the residual signal of an harmonic model of a tenor singer. The signal contains strong vibrato, and therefore, the bias due to the non-stationary parameters is expected to be significant. The harmonic models contain a maximum of 30 sinusoids at each time instant. We calculate the variance of the residual signal for the QIFFT, *DEH*, *DSH*, and *ASH* methods for a signal window of 800 samples and a FFT size of 4096 samples. The variance of the residual signal is compared to the QIFFT estimator and the reduction of the residual energy in different frequency bands that can be achieved with each estimator is listed in table (1).

From table (1) we can conclude that all bias reduction methods achieve significant improvements of the residual energy. It is interesting to compare the performance in the different frequency bands. In the low band the improvement is in the range from 3-4dB. The improvement is less pronounced because the FM modulation extend is low. In the mid band range the FM modulation becomes stronger and the reduction methods achieve residual energy reduction from 7.3-9.3dB. For the highest band the FM modulation is still stronger, but the noise level is higher as well such that the reduction of the residual energy is not as strong.

The advantage of the demodulation methods over *ASH* is clearly visible. The *DEX* estimator improves the reduction of the *ASH* estimator by 0.5-1.2dB. The *DSX* estimator is clearly the best with an improvement compared to the *ASH* estimator by 0.8-2dB. The residual signals for the QIFFT and *DSH* estimator are shown in figure Figure 3. The reduction of the residual is clearly visible.

5. CONCLUSIONS

In the present paper we have shown that an efficient bias reduction strategy for estimation of sinusoidal parameters consists of a frequency slope estimation and demodulation prior to application of the standard QIFFT estimator. The procedure significantly reduces the bias of the standard estimator. It does not require the use of a Gaussian analysis window and does work for a much larger range of modulation depths than a recently proposed algorithm. The computational costs are significantly higher then those for the standard estimator ($\approx$ factor 8). However, they are sufficiently low such that real time estimation of some tenth of sinusoids from audio signals can be achieved. By means of investigation into the reduction of the residual energy that can be obtained for a real world vibrato signal we have shown that the proposed enhanced demodulation estimator is effectively working in real world situations. It has been shown that compared to the standard QIFFT estimator the reduction of the residual error depends on the frequency range and can be as large as 6-9dB.

6. REFERENCES

[1] X. Amatriain, J. Bonada, A. Loscos, and X. Serra, "Spectral processing," in *Digital Audiuo Effects*, U. Zölzer, Ed., chapter 10, pp. 373–438. John Wiley & Sons, 2002.

[2] T. F. Quatieri and R. J. McAulay, "Speech transformation based on a sinusoidal representation," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 34, no. 6, pp. 1449–1464, 1986.

[3] M. Abe and J. O. Smith, "AM/FM rate estimation for time-varying sinusoidal modeling," in *Proc. Int. Conf. on Acoustics, Speech and Signal Processing*, 2005, pp. 201–204 (Vol. III).

[4] T. Abatzoglou, "Fast maximum likelihood joint estimation of frequency and frequency rate," in *ICASSP*, 1986, pp. 1409–1412 VOL. II.

[5] S. Saha and S. M. Kay, "Maximum likelihood parameter estimation of superimposed chirps using monte carlo importance sampling," *IEEE Trans on Signal Proc.*, vol. 50, no. 2, pp. pp. 224–230, 2002.

[6] J. S. Marques and L. B. Almeida, "A background for sinusoid based representation of voiced speech," in *Proc. Int. Conf. on Acoustics, Speech and Signal Processing*, 1986, pp. 1233–1236.

[7] G. Peeters and X. Rodet, "SINOLA: A new analysis/synthesis method using spectrum peak shape distortion, phase and reassigned spectrum," in *Proc. Int. Computer Music Conference*, 1999, pp. 153–156.

[8] M. Abe and J. O. Smith, "Design criteria for the quadratically interpolated FFT method (I): Bias due to interpolation," Tech. Rep. STAN-M-117, Stanford University, Department of Music, 2004, available at `http://ccrma.stanford.edu/STANM/stanms/stanm114/index.html`.

[9] G. Peeters, *Modèles et modification du signal sonore adapté à ses charactéristiques locales*, Ph.D. thesis, Univertsité Paris 6, 2001, available at `http://recherche.ircam.fr/equipes/analyse-synthese/peeters/ARTICLES/Peeters_2001_PhDThesisv1.1.pdf`, french only.

[10] A. Röbel, "Estimation of partial parameters for non stationary sinusoids," in *Proc. Int. Computer Music Conference (ICMC)*, 2006.

[11] B. Ristic and B. Boashash, "Comments on "The Cramer-Rao lower bounds for signals with constant amplitude and polynomial phase"," *IEEE Transactions on Signal Processing*, vol. 46, no. 6, pp. 1708–1709, 1998.

[12] A. Röbel, "A new approach to transient processing in the phase vocoder," in *Proc. of the 6th Int. Conf. on Digital Audio Effects (DAFx03)*, 2003, pp. 344–349.

REALTIME MULTIPLE-PITCH AND MULTIPLE-INSTRUMENT RECOGNITION FOR MUSIC SIGNALS USING SPARSE NON-NEGATIVE CONSTRAINTS

Arshia Cont

Ircam-CNRS UMR 9912, IMTR Team,
Paris, France. and
Music Department, UCSD, La Jolla, CA.
cont@ircam.fr

Shlomo Dubnov

Music Department,
University of California in San Diego
La Jolla, CA.
sdubnov@ucsd.edu

David Wessel

Center for New Music and Audio Technologies,
University of California in Berkeley
Berkeley, CA.
wessel@cnmat.berkeley.edu

ABSTRACT

In this paper we introduce a simple and fast method for realtime recognition of multiple pitches produced by multiple musical instruments. Our proposed method is based on two important facts: (1) that timbral information of any instrument is pitch-dependant and (2) that the modulation spectrum of the same pitch seems to result into a persistent representation of the characteristics of the instrumental family. Using these basic facts, we construct a learning algorithm to obtain pitch templates of all possible notes on various instruments and then devise an online algorithm to decompose a realtime audio buffer using the learned templates. The learning and decomposition proposed here are inspired by non-negative matrix factorization methods but differ by introduction of an explicit sparsity control. Our test results show promising recognition rates for a realtime system on real music recordings. We discuss further improvements that can be made over the proposed system.

1. INTRODUCTION

We address two important problems often discussed in the music information retrieval and computer music research communities: estimating multiple fundamental frequencies of music signals and musical instrument recognition. Both topics have received substantial effort from the research community especially in the recent years for polyphonic sounds (as opposed to solo or monophonic audio). Both are also important tasks for many applications including automatic music transcription, music information retrieval and computational auditory scene analysis. Another motivation for this work is the continuing need for live algorithms in computer music where the recognition of musical characteristics of the signal such as pitches and instruments becomes essential.

The multiple-pitch detection literature contains a wide variety of methods spanning from pure signal processing models to machine learning methods for both music and speech signals. For an excellent overview of different methods for multiple-$f0$ estimation, we refer the reader to [1]. The main aim of instrument identification is to determine the number and the names of the instruments present in a given musical excerpt. Whereas musical instrument recognition studies mainly deals with solo musical sounds, the number of those dealing with polyphonic music has been increasing in the recent years. In [2], Kashino et al. develop a template-matching method that compares the observed waveform locally with sum of template waveforms that are phase aligned, scaled, and filtered adaptively. Similarly [2, 3] use feature matching methods where features computed in zones where several notes overlap are modified or discarded before stream validation depending on their type. Other systems directly address the instrument identification without considering note models or pitch detection. In [4], Essid et al. introduce an SVM model with a hierarchical taxonomy of a musical ensemble that can classify possible combinations of instruments played simultaneously. In another approach, Livshin and Rodet [5] use an extensive set of feature descriptors on a large set of pitched instrument sound samples, reducing the feature dimensions with Linear Discriminant Analysis and then classifying the sounds with a KNN method. More recently Kitahara et al. [6] have proposed a method for visualizing the instrument existence probabilities in different frequency regions.

In this paper, we propose a new technique that recognizes multiple pitches along with their instrument origin in polyphonic musical audio signals and in realtime; hence, addressing both problems mentioned earlier. The main difference between our proposed method and the ones discussed above is the fact that our system is geared towards real-time recognition and for realistic musical situations. Our approach is similar to [2] where instrument-based pitch templates are being matched to the ongoing audio but differs significantly by the extensive reliance on sparse machine learning in our approach. Our proposal is inspired by simple observational facts regarding the nature of musical instruments and consists of decomposing an ongoing audio signal using previously learned instrument-dependant pitch templates and sparse non-negative constraints. We discuss the basic idea and general architecture of the algorithm in section 2. The algorithm both in learning and realtime decomposition phases, uses a recently introduced signal representation scheme based on modulation spectrum [7]. The key fact here is that the modulation spectrum of musical instruments seems to be an important discriminating factor among them. We will discuss the modulation spectrum and its pertinence to our problem in

section 3 as the main signal processing front-end for our algorithm. In section 4 we show how instrument templates are learned. This learning is once-for-all and is based on *Non-negative Matrix Factorization* (NMF) algorithm [8]. These learned templates would be imported to the main algorithm for realtime instrument-based pitch detection called sparse non-negative decomposition detailed in section 5. This is followed by some results and discussion on further improvement envisioned for the proposed system. An earlier version of the machine learning algorithm proposed here has appeared previously in [9] by the first author and for a different application. In this paper, we have refined the learning methods and are introducing it in a more elaborate and different context.

2. GENERAL ARCHITECTURE

As mentioned earlier, we attempt to address both the problem of multiple-pitch detection and musical instrument recognition. The motivation behind this mix is the simple fact that for each given musical instrument, the timbral profiles vary along different pitches or notes produced by the instrument. Moreover, the timbral profile of a given pitch on a given instrument varies along different modes of performance for certain instruments (for example playing *ordinario* or *pizzicato* on violin family). Given this fact, we propose learning *templates* for each sound produced in each instrument once and for all, and use these templates during realtime detection.

Another important motivation behind the proposed algorithm is the simple intuition that humans tend to use a reconstructive scheme during detection of multiple pitches or multiple instruments and based on their history of timbral familiarity and music education. That is to say, in music dictation practices, well-trained musicians tend to transcribe music by conscious (or unconscious) addition of familiar pitches produced by musical instruments. The main idea here is that during detection of musical pitches and instruments, there is no direct assumption of *independence* associated with familiar patterns used for reconstruction and we rely more on *reconstruction* using superpositions.

Considering these facts, we can generally formulate our problem by *non-negative* factors. Non-negativity in this case simply means that we do not *subtract* pitch patterns in order to determine the correct combination but rather, we somehow manage to directly point to the correct combination of patterns that reconstruct the target by simple linear superposition. Mathematically speaking, given V as a non-negative representational scheme of the realtime audio signal in $\mathbb{R}_+^N$, we would like to achieve

$$V \approx WH \tag{1}$$

where W is a non-negative $\mathbb{R}_+^{N\times r}$ matrix holding r templates corresponding to objects to be detected and H is a simple non-negative $r \times 1$ vector holding the contribution of each template in W for reconstructing V. During realtime detection, we are already in possession of W and we tend to obtain H indicating the presence of each template in the audio buffer that is arriving online to the system in V. Given this formulation, there are three main issues to be addressed:

1. What is an efficient and pertinent representation for V?
2. How to learn templates in W using this representation?
3. And how to obtain an acceptable result in H in realtime?

We will give a general overview of the three questions above in the following subsections and present algorithmic descriptions in the coming sections.

2.1. Representational Front-end

Any representational front-end chosen for the formulation above, should at least meet two important properties: (1) obviously it should have enough information for discrimination between instruments, and (2) due to the non-negative formulation in equation 1 it should preserve itself when multiple instruments are present at least to a good extent and in our case, observe superposition of different instruments.

Dubnov et al. have shown in [10] that phase coupling is an important characteristic of a sustained portion of sound of individual musical instruments and show results obtained for various instruments observing consistencies in phase coupling templates for at least flute and cello. Furthermore, they note that the statistical properties of a signal due to phase variation can not be easily revealed by standard spectral analysis techniques due to the fact that second-order statistics and the power spectrum are *phase blind*. In their proposal they use a quadratic phase coupling method using higher-order statistics to obtain the phase coupling representation. Using additive sinusoidal analysis, their method is highly sensitive to the fundamental frequencies of the sound itself.

The representational front-end we propose in this paper is inspired by findings in [10] as an indirect but efficient method to represent spectral modulations of the signal, also capable of representing pitch information. We will detail this representational scheme in section 3.

2.2. Sparsity of the solution

Equation 1 simply assumes a linear combination of the previously learned templates with non-negative coefficients for reconstruction of V or learning H. The price to pay for this simplicity is of course solving for the correct results in H where there are many possible combinations of templates that might achieve a given error criterion. This issue becomes even more important if there is no mathematical independence between the basis stored in W as templates. This is a major difficulty with non-negative constraint problem solving. More specifically, for our problem, harmonic relations between pitches of an instrument and among instruments themselves always lead to various approximate solutions for H and leading to the famous *octave errors* and more.

To overcome this problem, we use the strong assumption that the correct solution for a given spectrum (in V) uses a minimum of templates in W, or in other words, the solution has the minimum number of non-zero elements in H. This assumption is hard to verify for every music instrument and highly depends on the template representations in W, but is easily imaginable as harmonic structure of a music note can be minimally expressed (in the mean squared sense) using the original note than a combination of its octaves and dominant.

Fortunately, this assumption has been heavily studied in the field of *sparse coding*. The concept of sparse coding refers to a representational scheme where only a few units out of a large population are effectively used to represent typical data vectors [11]. In section 5 we propose a technique to control sparsity in a non-negative constraint problem.

3. MODULATION SPECTRUM

For non-stationary signal classification, features are traditionally extracted from a time-shifted, yet short data window. For instrument classification, these short-term features do not efficiently capture or represent longer term signal variations important for the given task and can barely represent important discriminative characteristics such as spectral envelope or phase coupling for musical instrument recognition. Sukittanon et al. in [7] propose a modulation spectrum representation that not only contains short-term information about the signal, but also provides long-term information representing patterns of time variation on the spectrum itself. In this model, the audio signal is the product of a narrow bandwidth, stationary low-pass modulating random process $m(t)$ and the high-pass carrier, a deterministic function $c(t)$

$$x(t) = m(t) \cdot c(t)$$

For the model to be accurate, $m(t)$ is assumed to be non-negative and its bandwidth does not overlap with that of $c(t)$. The above model has been applied to speech and audio coding [12]. Following the observations in section 2.1 and in [10], we study the feasibility of this representation for our task and hope that $m(t)$ will provide an informative representation for pitched musical instruments.

Modulation Spectrum is based on a two-dimensional representation of the acoustic and modulation frequency or a joint frequency representation. Moreover, it does not require prior estimate of the periodicity of the signal. One possible representation of this form is $P_x(\eta, \omega)$, as a transform in time of a demodulated short-time spectral estimate where ω and η are *acoustic frequency* and *modulation frequency* respectively. To obtain this representation, we first use a spectrogram with an appropriately chosen window length to estimate a joint time-frequency representation of the signal $P_x(t, \omega)$. Second, another transform (Fourier in our case) is applied along the time dimension of the spectrogram to estimate $P_x(\eta, \omega)$. Figure 1 shows the analysis structure undertaken on the audio (top) to obtain the modulation spectrum (down). A more rigorous view of $P_x(\eta, \omega)$ is the convolution in ω and multiplication in η of the correlation function of a Fourier transform of the signal $x(t)$ and the underlying data analysis window $w(t)$, as in equation 2 [7].

$$\begin{aligned} P_x(\eta, \omega) &= \left(W^*(\omega - \frac{\eta}{2})W(\omega + \frac{\eta}{2})\right) \\ &* \; w\left(X^*(\omega - \frac{\eta}{2})X(\omega + \frac{\eta}{2})\right) \end{aligned} \tag{2}$$

Figure 2 shows this representation for one analysis frame of piano and trumpet both playing A_4 ($f_0 \approx 440Hz$). The time-span of this analysis corresponds to the length of the first transform N_1, the length of the second transform N_2 and the sampling frequency f_s which here are 2048, 32, 44100 leading to a span of almost 1.5 seconds. Frequency modulation resolution, similar to frequency and time resolution in Fourier transform, relies on the choice of N_2 and both transforms' overlap size H_1 and H_2.

Interpretation of $P_x(\eta, \omega)$ above is straightforward. The values of $P_x(\eta, \omega)$ lying along $\eta = 0$ is an estimate of the non-stationary information about the signal which, in our case, corresponds mostly to harmonic partials in the spectrum. Values along $\eta > 0$ correspond to the degree of spectral modulation. For example, in figure 2 almost all partials of the trumpet are being modulated whereas for piano (left) modulation decreases for higher partials.

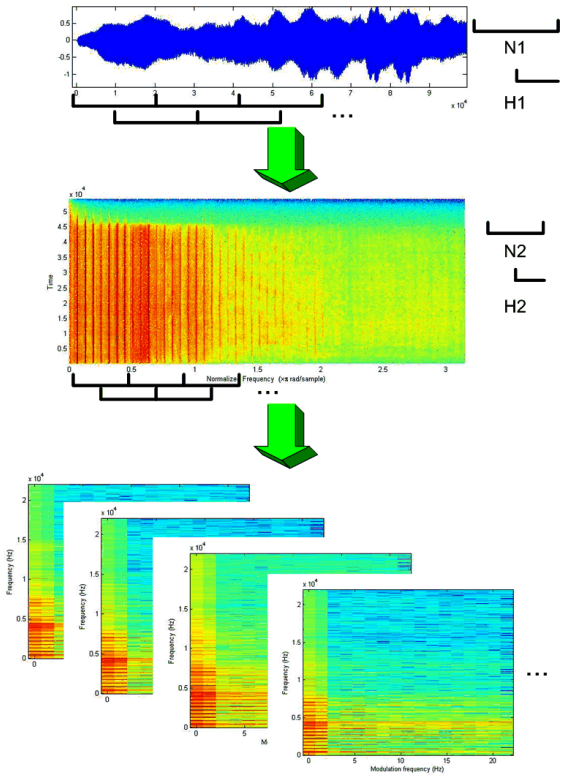


Figure 1: *Analysis structure for obtaining modulation spectrum (bottom) from audio (top).*

The non-negativity of the modulation spectrum representation and its ability to demonstrate phase coupling of instruments as modulation frequencies makes it a perfect candidate for the representation needed for V in our problem definition. Furthermore, Atlas et al. discuss associativity of this representation in [13], leading to superposition of instrument templates when several are present in the spectrum. This is demonstrated in figure 3 where modulation spectrum of flute playing A_6 alone is represented at left and modulation spectrum of a recording of piano playing A_4 and flute playing A_6 at the same time is represented at the right. Intuitively, the figure on the right of figure 3 would be a straight superposition of the left figures in figure 2 and 3 despite their different scaling.

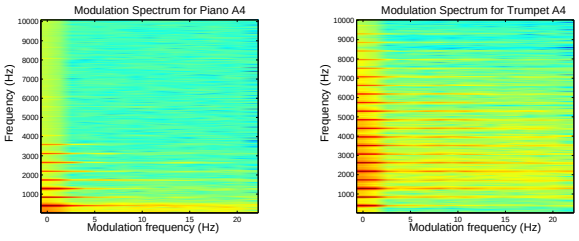


Figure 2: *Modulation Spectrum of Piano (left) and Trumpet (right) playing $A4$, zoomed over $0 - 10KHz$ acoustic frequencies.*

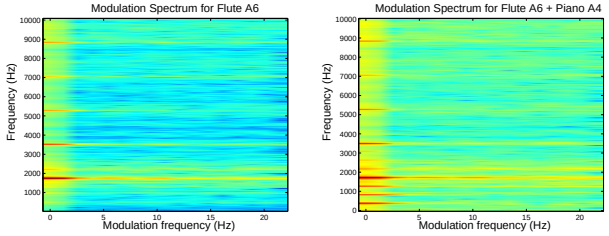


Figure 3: *Modulation Spectrum of Flute playing A_6 (left) and Piano (A_4) and Flute (A_6) playing together (right), zoomed over $0-10KHz$ acoustic frequencies.*

Hence, we adopt this two-dimensional joint frequency representation as the front-end of our system.

4. LEARNING INSTRUMENT-BASED PITCH TEMPLATES

As mentioned in section 2, the proposed system solves for the existence of previously learned instrument-based pitch templates (stored in W in our notation). Here we discuss how these templates are learned and resolve the second question in section 2. As a reminder, W contains modulation structures of all pitches of each given instrument. For example, for an acoustic piano, matrix W would contain all 88 notes as 88 different 2-D representations. To this end, training is done on databases of instrumental sounds [14, 15] using an off-line training algorithm that learns different modulation structures of instruments by browsing all sounds given in the database and stores them in matrix W for future use.

For each audio file in the database, training is an iterative NMF algorithm [8] with a symmetric kullback-leibler divergence for reconstruction error as shown in Equation 3, where $\otimes$ is an element by element multiplication. In this off-line training, V would be the modulation spectrum of the whole audio file as described in Section 3 and the learning algorithm factorizes V as $V \approx WH$. The subscript a refers to the a^{th} template and other subscripts in equation 3 are vector indexes used during learning. In order to obtain precise and discriminative templates, we put some constraints on W vectors learned during each NMF iteration. For each sound in the database (or each pitch) we force the algorithm to decompose V into two objects (W has two 2-D elements) where we only learn one vector and have the other fixed as white non-negative noise, where only the first one would be stored for the global W. This method helps the algorithm focus more on the harmonic and modulation structure of V. Furthermore, we require modulation frequencies higher that zero ($\eta > 0$) at each iteration by a constant factor ($Emph$ in equation 3). The idea behind this factor is to emphasize non-stationary structure of the signal, important for between instrument discrimination.

$$
\begin{aligned}
H_{a\mu} &\longleftarrow Emph \otimes H_{a\mu} \frac{\sum_i W_{ia} V_{i\mu}/(WH)_{i\mu}}{\sum_k W_{ka}} \\
W_{ia} &\longleftarrow Emph \otimes W_{ia} \frac{\sum_i H_{a\mu} V_{i\mu}/(WH)_{i\mu}}{\sum_\nu H_{a\nu}}
\end{aligned}
\tag{3}
$$

When the training reaches an acceptable stopping criteria, the modulation spectra in the local W will be saved in the global W and the algorithm continues to the next audio file in the database until it constructs W for all given sounds in the database. Figure 4 shows learned modulation spectrum templates for flute and violin playing A_4. During analysis, the parameters are $N_1 = 2048, N_2 = 32, H_1 = 1024$, and $H_2 = 16$ leading to a time resolution of $\sim 370ms$ and modulation upper bound of around $21Hz$ for a sampling frequency of $44100Hz$. Both templates were trained on audio files in the SOL database [15] and were converged after slightly more than 100 iterations.

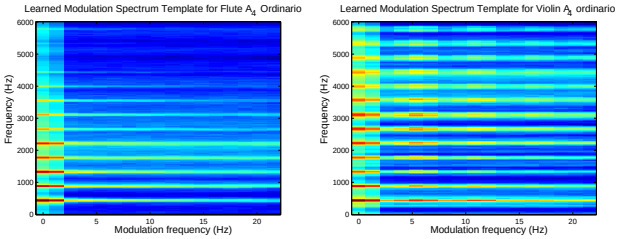


Figure 4: *Learned Modulation Spectrum of Flute A_4 (left) and Violin A_4 (right).*

Note that using this type of representation for templates has an important disadvantage. The modulation spectrum described above provides a large dimension compared to traditional short-term spectral estimations. To compensate for this, we reduce the representation by cutting frequencies above $6KHz$. This choice was adopted by trial-and-error and since most useful partial and modulation information lie below this threshold. Moreover, during learning and decomposition, we consider the 2-D modulation spectrum and templates as *images* that hence, can be reshaped into a 1-D vector and back.

5. SPARSE NON-NEGATIVE DECOMPOSITION

We are now in a position to address the third and last issue brought in section 2. Having V as the modulation spectrum analysis of real-time audio and W as stored instrument-based pitch templates, we would like to obtain H such that $V \approx WH$. As mentioned earlier in section 2.2, in order to decompose the spectrum using learned pitch templates, the solution needs to be sparse. One of the useful properties of the original NMF [8] is that it usually produces a sparse representation of the data. However this sparseness is more of a side-effect than a goal and one can not control the degree to which the representation is sparse. In this section, we introduce a modified sparse non-negative decomposition algorithm.

Numerous sparseness measures have been proposed and used in the past. In general, these measures are mappings from $\mathbb{R}^n$ to $\mathbb{R}$ which quantify how much energy of a vector is packed into a few components. As argued in [16], the choice of sparseness measure is not a minor detail but has far reaching implications on the structure of a solution. Very recently, Hoyer has proposed an NMF with sparseness constraints by projecting results into ℓ_1 and ℓ_2 norm-spaces [17]. Due to real-time considerations and the nature of sparseness in audio signals for pitch determination we propose a modified version of Hoyer's method described in [17].

The definition commonly given for sparseness is based on the ℓ_0 norm defined as the number of non-zero elements

$$\|X\|_0 = \frac{\#\{j, x_j \neq 0\}}{N}$$

where N is the dimension of vector X. It is a characteristic for the ℓ_0 norm that the magnitude of non-zero elements is ignored. Moreover, this measure is only good for noiseless cases and adding a very small measurement noise makes completely sparse data completely non-sparse. A common way to take the noise into account is to use the ℓ_ϵ norm defined as follows:

$$\|X\|_{0,\epsilon} = \frac{\#\{j, |x_j| \geq \epsilon\}}{N}$$

where parameter ϵ depends on the noise variance. In practice, there is no known way of determining this noise variance which is independent of the variance in x. Another problem of this norm is that it is non-differentiable and thus can not be optimized with gradient methods. A solution is to approximate the ℓ_ϵ norm by $\tanh$ function,

$$g(x) = \tanh(|ax|^b)$$

where a and b are positive constants. In order to imitate ℓ_ϵ norm, the value of b must be greater than 1.

In addition to the $\tanh$ norm, we force an ℓ_2 constraint on the signal. This second constraint is crucial for the normalization of the results and emphasis on significance of factorization during note events in contrary to silent states.

In summary, the sparseness measure proposed is based on the relationship between the ℓ_ϵ norm and the ℓ_2 norm as demonstrated mathematically in Equation 4.

$$\text{sparseness(x)} = \frac{\sqrt{N} - \sum \tanh(|x_i|^2)/\sqrt{\sum x_i^2}}{\sqrt{N} - 1} \qquad (4)$$

Algorithmic realization of this sparsity constraint is a straightforward and cheap iterative procedure that projects the results first to the ℓ_ϵ hyperplane and then solves for the intersection of this projection with the hyperplane possessed by ℓ_2. Figure 5 shows a synthetic signal (left) and its sparse projection using the proposed procedure with $\ell_\epsilon = 0.9$ and ℓ_2 equal to signal's energy.

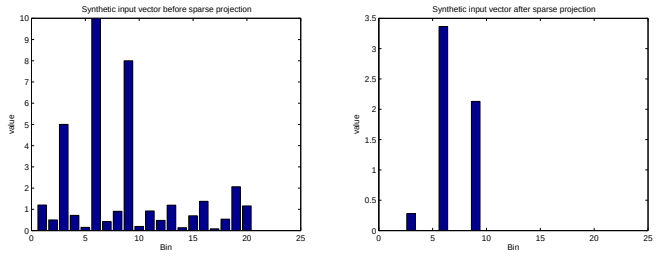


Figure 5: *Synthetic signal before sparse projection (left) and after (right).*

For non-negative sparse decomposition, we use gradient descent updates instead of the original NMF multiplicative updates (Equation 3) and project each vector in real-time to be non-negative and have desired ℓ_2 and ℓ_ϵ norms. This projected gradient descent, adapted from [17], is outlined below. Once again this algorithm shows the factorization for H when templates are known.

Given V and W, find the non-negative vector H with a given ℓ_ϵ norm and ℓ_2 norm:

1. Initialize H to random positive matrices or to previous value of H in sequence
2. Iterate
   (a) Set $H = H - \mu_H W^T(WH - V)$
   (b) Set $s_i = h_i + (\ell_\epsilon - \sum \tanh(h_i^2))/N$ and $m_i = \ell_\epsilon / N$
   (c) Set $s = m + \alpha(s - m)$ where $\alpha = \frac{-(s-m)^T m + \sqrt{((s-m)^T m)^2 - \sum(s-m)^2(\sum m^2 - \ell_2^2)}}{\sum(s-m)^2}$
   (d) Set negative components of s to zero and set $H = s$

Algorithm 1. Sparse Non-negative Matrix Decomposition

Here, step (a) is a negative gradient descent and (b) through (d) are the projection process on the ℓ_ϵ and ℓ_2 space. In (b) we are projecting the vector to the ℓ_ϵ hyperplane and (c) solves a quadratic equation ensuring that the projection has the desired ℓ_2 norm.

For realtime pitch/instrument detection, the ℓ_2 norm is provided by the spectrum energy of the realtime signal (directly calculated from the column in V corresponding to $\eta = 0$) and the ℓ_ϵ takes values between 0 and 1, is user-specified and can be controlled dynamically. The higher the ℓ_ϵ, the more sparse is the solution in H. V would be a vector of size $N_1 \times N_2$ where here we use $N_1 = 2048$ and $N_2 = 32$ and further reduced (along N_1) to capture modulation structures up to about $6KHz$ acoustic frequency in a sampling rate of $44.1KHz$. Equivalently, W would be a matrix of SizeOf$(V) \times m$ with m as the number of templates and H would be a vector of size m.

6. EVALUATION

A clean evaluation of a systems such as the one proposed in this paper bears practical difficulties. It should be clear by now that we are attempting towards a multi-instrument transcription of music signals in form of a *piano roll* presentation. To evaluate such representation one needs an annotated and transcribed music of the same type to an order of milli-seconds. There has been recent attempts in creating such database but for monophonic music or in the best case, polyphonic but mono-instrument sounds (such as piano music). Evaluation procedures that has been undertaken so far in the literature do not seem to be close to ideal either. In systems where the authors aim for multiple-instrument identification, the pitch information is missing [4, 6]. Otherwise, other researchers aimed at manual mixing of single note recordings of different instruments as the basis of their evaluations (for example in [5]).

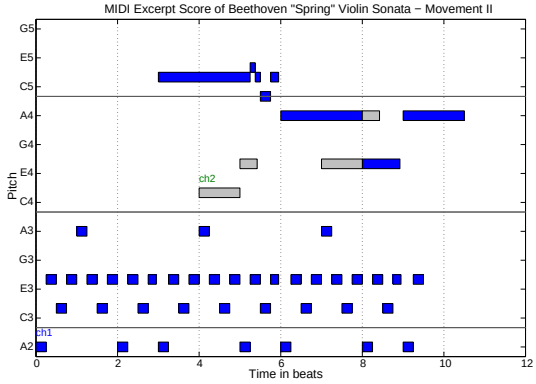
As mentioned in the beginning of this paper, our system is destined towards real-time applications in computer music systems. Therefore, it is vital that the evaluation procedure is done on real music recordings and in real musical situations despite the difficulties of such approach.

In this section we showcase the performance of our system in two manners: (1) A subjective evaluation where we demonstrate the real-time output of the system on short musical examples and compare the results visually with the piano-roll representations of their scores. (2) An objective evaluation where we evaluate the algorithm on mixed music recordings and provide detailed results.

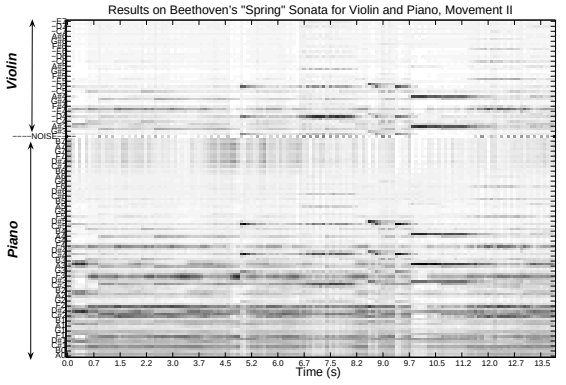
All audio files and results used during evaluation as well as more fine-tuned and detailed images can be viewed on the project's website[1].

6.1. Subjective Evaluation

Figures 6(a) and 6(b) show the performance of the system (bottom) on a real recording of the first phrase of Beethoven's Sonata for Piano and Violin *(The spring)*. A piano roll representation of the MIDI score of the same phrase is represented in figure 6(a) where the Piano section has darker color than the violin part. In the sample result (figure 6(b)), decomposition results are represented as an image where the Piano and Violin results occupy a separate space. The Y-axis represents pitches for each instrument (88 for Piano and 41 for Violin) and are sorted in ascending order to resemble a piano-roll representation.



(a) MIDI Score Representation

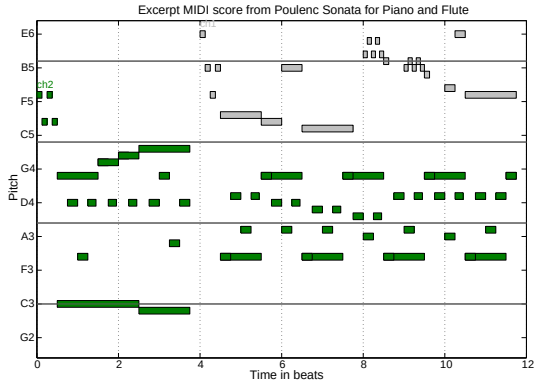


(b) Decomposition Results

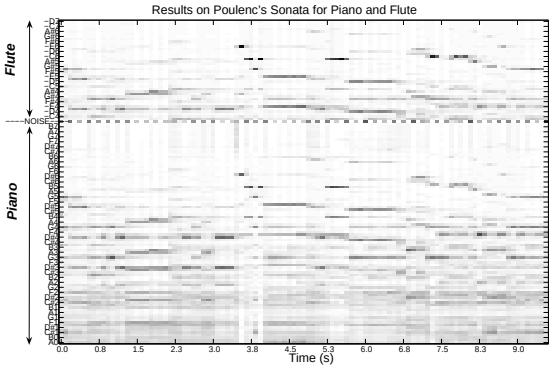
Figure 6: *Sample result (1): Beethoven's* Spring *Violin-Piano Sonata, 2nd movement, starting bars with score (top) and system result (bottom).*

Similarly, figures 6.1 shows the performance of the system (bottom) on a real recording of a few bars from Francis Poulenc's Sonata for Flute and Piano with the score excerpt shown in a piano roll presentation on the top. Here again, the flute section is represented by a lighter color in the piano roll score of figure 7(a).

[1] `http://cosmal.ucsd.edu/arshia/DAFx07/`



(a) MIDI Score Representation



(b) Decomposition Results

Figure 7: *Sample result (2): Francis Poulenc's Sonata for Flute and Piano (excerpts).*

The parameters used for training and real-time decompositions for both examples are as follows: $F_s = 44100Hz, N_1 = 4096, N_2 = 32, H_1 = 256$, and $H_2 = 16$. During real-time analysis, these choices leave us with an analysis time-span of almost 3 seconds and response delay of 93 milli-seconds.

For a subjective evaluation, it suffices to compare the score piano-roll representation with the result images in figures 6(b) and 7(b). For the Piano parts, specially for the Piano and Violin example, the low notes are hard to distinguish, especially with the current scaling of the paper. However, in both figures the main contour of both scores can be easily detected with the eyes. An important remark here is the (weaker) presence of the first instruments (Violin and Flute) in the accompaniment instrument (Piano). Both detailed analysis of the results and the confusions are addressed in the next section.

6.2. Objective Evaluation

Due to the lack of high-resolution annotated and transcribed ensemble recordings, we tend to mix transcribed and annotated monophonic music for different instruments and evaluate the performance of our system on the mixed audio. The advantage of this approach is that first, we will be dealing with real music recordings and two, we can easily calculate precisions for instrument/pitch

detection across instruments since the annotations for each instrument are separate. The disadvantage, of course, is that after-the-fact mixing of two instruments can not demonstrate eventual spectral fusions common in ensemble recordings (which was not the case in our subjective evaluation). For this paper, we focus on two-instrument mixes and address more enhanced evaluations in another publication.

Audio and annotation files used for this evaluation session are taken from the *Score Following Evaluation Task* prepared by the author for MIREX 2006 [18] and also from a previous experiment reported in [9]. Table 1 shows the specification of Audio and (aligned) MIDI files used during the evaluation. The MIDI annotations that come with each audio file, provide aligned score to audio information. Note that although these annotation were created automatically and double checked using a high-resolution analysis software, they are not perfect especially in the case of Piano because of the difficulty in assigning correct note lengths in a polyphonic situation and in the presence of the piano pedal. This issue is quite present for piece number 2 which is usually played with a high utilization of the sustain pedal.

Table 1: Specification of Audio and Midi used for evaluation

#	Piece Name	Time	Events	Instr.
1	Mozart's *Piano Sonata in A major,K.331*	9:55	4268	Piano
2	Chopin's *Nocturne no.2, opus 9*	3:57	1291	Piano
3	Bach's *Violin Sonata 1, Movement 4*	3:40	1622	Violin
4	Bach's *Violin Sonata 2, Movement 4*	5:13	2042	Violin

For this evaluation we created two Piano and Violin mixtures according to their lengths: $1+4$ and $2+3$, and ran the system on both mixtures. Mixing starts at time zero so since the piano recordings are always longer in our case, we are sure that during the length of the violin parts there is always activity in both instruments and we are left with some extra piano-only section in the end. During evaluations, for each note event in the aligned score, we look at the corresponding frames of the analysis observation and check if the corresponding template has high activity and if it is among the top N templates, where N specifies the number of pitches active at the event frame time taken out of the reference MIDI. This way, for each event in the score we can have a precision percentage and the overall mean of these can represent the algorithm's precision. Moreover, since we do not have any specific temporal model and also the ending of notes are usually doubtful (especially for Piano) we can consider (subjectively) positive detection during at least 80% of a note life to be *acceptable* and refine the precision. Cross classification can be computed in the same way by switching the piano and violin references between results.

Tables 2 and 3 show confusion matrices out of the above evaluation for each mix (where numbers refer to specifications in table 1). This confusion matrix is to be read as follows: the row elements correspond to the results being evaluated and the column elements correspond to the reference alignment being used for evaluation. For example, element $(1,2)$ refers to the percentage within which the system has decoded violin elements in the Piano results. Therefore, it is natural that this confusion matrix is not symmetrical. On another note, values in the confusion matrices do not add up to 100%. Each row column of the matrix represents the instances in a predicted instrument class while each row represents the instances in an actual class. These measures obviously are not representative of all sorts of errors a transcription system can undergo. For a detailed description of different kinds of errors in a music transcription problem we refer the reader to [19]. In what follows we emphasize the *precision* rate and inner-instrumental confusion thereof through the results. Finally, note that precision rates in Tables 2 and 3 correspond to both (multiple) pitch and instrument classification where the reference for both is obtained from the aligned MIDI scores to audio.

Results in tables 2 and 3 suggest that the precision rate (diagonal values) for the violin parts are significantly higher than the Piano part. This is mainly due to the fact that the Violin sections (files 3 and 4 in table 1) are much louder than the piano audio files and we did not normalize the loudness before mixing to be as natural as possible. Other reasons for the deficiency of the Piano pitch/instrument detection comes from the fact that in both pieces there is an extensive use of sustain pedals which confuses the system when trying to match templates for reconstructing the ongoing modulation structure. Furthermore, lower Piano precision in Table 2 is because the sustain pedal is being used much more in Chopin's *Nocturne* than Mozart's *Sonata*, which is stylistically reasonable.

Confusion Matrices

	Piano	**Violin**
Piano	45%	9.2%
Violin	17.1%	67.5%

Table 2: Mix of $2+3$

	Piano	**Violin**
Piano	52.8%	17.9%
Violin	24.3%	89.3%

Table 3: Mix of $1+4$

Overall, given the nature of the problem, that is simultaneous multiple-pitch and multiple-instrument detection in real-time, the results are satisfactory and not far from other state-of-the-art systems cited earlier in section 1.

7. CONCLUSION AND DISCUSSION

In this paper we presented a technique for detection of multiple-pitches produced by multiple-instruments and in real-time. The core of the proposed system relies on a rather simple machine learning principle based on sparse non-negative constraints. The simplicity behind this algorithm is due to observations on the nature of musical instruments and basic facts regarding musical pitch and timbre structures. After formulating the problem we discussed three main issues regarding the formulation and presented solutions for each.

If the proposed method is to be useful in computer music applications, the precision rates should obviously be higher than the ones in Tables 2 and 3. The work presented in this paper is regarded as a first step towards the complex problem of multiple pitch and instrument recognition in real-time. However, obtained results with a more rigorous evaluation framework as stated before, are close to the state-of-the-arts reported elsewhere. Here we elaborate on the future directions of this project and on how the proposed algorithm can be improved.

The on-line learning algorithm developed in section 5, uses a simple gradient descent update that is projected at each iteration to assure sparsity. From a machine learning perspective, gradient descent updates are not always the best solution and more intelligent optimization techniques such as convex optimization and semi-definite programming would be more suitable. However, for this experiment and due to our real-time constraints, we

adopted the gradient descent approach and will report on more advanced methods in later publications. Also, the sparse constraints introduced are quite powerful in order to avoid inner-instrumental overuse of templates but does not directly address avoiding inner-instrumental confusions. New sparsity measures should be experimented in order to overcome this issue. On another note, we use the amplitude (or absolute value) of the complex modulation spectrum reported in section 3. Later improvements will consider the complex values or in other words, the phase information of the modulation spectrum directly into the decomposition algorithm.

We conducted subjective and objective evaluations of the algorithm but in the limited case of two instruments. A more elaborate evaluation procedure is needed to discover true deficiencies and outcomes of the proposed algorithm. However, for the given task, the evaluation frameworks that has been introduced so far in the literature do not provide sufficient and accurate data for such fine-tuned evaluation. We will be elaborating on this subject to further improve the test-bed that can lead to better frameworks and improved systems.

Finally, as mentioned earlier in the paper, one of the main motivations for this research is to provide real-time tools for the computer musicians and researchers with their growing need for real-time detection tools. The algorithm and application proposed in this paper is currently under development for MaxMSP[2] and Pure Data[3] real-time computer music environments and will soon be available for free download[4].

8. REFERENCES

[1] A. de Cheveigné, "Multiple f0 estimation," in *Computational Auditory Scene Analysis: Principles, Algorithms and Applications*, D.-L. Wang and G.J. Brown, Eds., pp. 45–72. IEEE Press / Wiley, 2006.

[2] Kunio Kashino and Hiroshi Murase, "A sound source identification system for ensemble music based on template adaptation and music stream extraction," *Speech Commun.*, vol. 27, no. 3-4, pp. 337–349, 1999.

[3] J. Eggink and G. J. Brown, "A missing feature approach to instrument identification in polyphonic music," in *IEEE ICASSP*. 2003, Hong Kong.

[4] Slim Essid, Gaël Richard, and Bertrand David, "Instrument recognition in polyphonic music based on automatic taxonomies.," *IEEE Transactions on Audio, Speech & Language Processing*, vol. 14, no. 1, pp. 68–80, 2006.

[5] Arie Livshin and Xavier Rodet, "The significance of the non-harmonic "noise" versus the harmonic series for musical instrument recognition," in *International Symposium on Music Information Retrieval (ISMIR)*, 2006.

[6] T. Kitahara, K. Komatani, T. Ogata, H.G. Okuno, and M. Goto, "A missing feature approach to instrument identification in polyphonic music," in *IEEE ICASSP*. 2006, Toulouse.

[7] Somsak Sukittanon, Les E. Atlas, and James W. Pitton, "Modulation-scale analysis for content identification," *IEEE Transactions on Signal Processing*, vol. 52, no. 10, pp. 3023–3035, 2004.

[8] Daniel D. Lee and H. Sebastian Seung, "Algorithms for non-negative matrix factorization," in *Advances in Neural Information Processing Systems 13*, Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, Eds. 2001, pp. 556–562, MIT Press.

[9] Arshia Cont, "Realtime multiple pitch observation using sparse non-negative constraints," in *International Symposium on Music Information Retrieval (ISMIR)*. October 2006, Victoria, CA.

[10] S. Dubnov and X. Rodet, "Investigation of phase coupling phenomena in sustained portion of musical instruments sound," *Acoustical Society of America Journal*, vol. 113, pp. 348–359, Jan. 2003.

[11] D. J. Field, *Neural Computation*, vol. 6, chapter What is the goal of sensory coding?, pp. 559–601, 1994.

[12] M.S. Vinton and L.E. Atlas, "Scalable and progressive audio codec," *IEEE ICASSP*, vol. 5, pp. 3277–3280, 2001.

[13] Les Atlas and Christiaan Janssen, "Coherent modulation spectral filtering for single-channel music source separation," in *IEEE International Conference in Acoustics and Speech Signal Processing (ICASSP)*, 2005.

[14] Masataka Goto, Hiroki Hashiguchi, Takuichi Nishimura, and Ryuichi Oka, "Rwc music database: Popular, classical and jazz music databases.," in *International Symposium on Music Information Retrieval (ISMIR)*, 2002.

[15] Guillaume Ballet, Riccardo Borghesi, Peter Hoffmann, and Fabien Lévy, "Studio online 3.0: An internet "killer application" for remote access to ircam sounds and processing tools," in *Journée d'Informatique Musicale (JIM)*, paris, 1999.

[16] Juha Karvanen and Andrzej Cichocki, "Measuring sparseness of noisy signals," in *ICA2003*, 2003.

[17] Patrik O. Hoyer, "Non-negative matrix factorization with sparseness constraints.," *Journal of Machine Learning Research*, vol. 5, pp. 1457–1469, 2004.

[18] Arshia Cont, Diemo Schwarz, Norbert Schnell, and Christopher Raphael, "Evaluation of real-time audio-to-score alignment," in *International Symposium on Music Information Retrieval (ISMIR)*. October 2007, Vienna, Austria.

[19] G. Poliner, D.P.W. Ellis, A.F. Ehmann, E. Gomez, S. Streich, and B. Ong, "Melody transcription from music audio: Approaches and evaluation," *IEEE Transactions on Audio, Speech & Language Processing*, vol. 15, no. 4, pp. 1247–1256, 2007.

[2] `http://www.cycling74.com/`

[3] `http://crca.ucsd.edu/~msp/software.html`

[4] For progress, see: `http://cosmal.ucsd.edu/arshia/DAFx07/`

MULTIPITCH ESTIMATION OF QUASI-HARMONIC SOUNDS IN COLORED NOISE

Valentin Emiya, Roland Badeau, Bertrand David

GET - Télécom Paris (ENST), CNRS LTCI
46, rue Barrault, 75634 Paris cedex 13, France
valentin.emiya@enst.fr

ABSTRACT

This paper proposes a new multipitch estimator based on a likelihood maximization principle. For each tone, a sinusoidal model is assumed with a colored, Moving-Average, background noise and an autoregressive spectral envelope for the overtones. A monopitch estimator is derived following a Weighted Maximum Likelihood principle and leads to find the fundamental frequency (F_0) which jointly maximally flattens the noise spectrum and the sinusoidal spectrum. The multipitch estimator is obtained by extending the method for jointly estimating multiple F_0's. An application to piano tones is presented, which takes into account the inharmonicity of the overtone series for this instrument.

1. INTRODUCTION

Multipitch estimation is a critical topic for many applications, both in the field of speech processing (*e.g.* prosody analysis) [1] and in the context of musical signal analysis (*e.g.* automatic transcription) [2, 3]. The challenge offered by the spectral interference of the overtones of simultaneous notes has been taken up by various methods, some aiming at detecting a periodicity in the signal [4] or in its spectrum [5] while others use a combination of both spectral and temporal cues [6, 7]. Recent trends in the task include estimation in a bayesian framework [8] or in a perceptually compliant context [7]. The technique proposed in this paper is based on a Weighted Maximum Likelihood (WML) principle and belongs to the spectral estimators category.

This paper is organized as follows. Section 2 introduces the Maximum Likelihood principle applied to the proposed signal model. Section 3 then details the adaptation of the theoretical method to the multipitch estimation task in the case of piano sounds. Experimental results are given in section 4. Finally, conclusions are presented in section 5.

The research leading to this paper was supported by the French GIP ANR under contract ANR-06-JCJC-0027-01, Décomposition en Éléments Sonores et Applications Musicales - DESAM, and by the French Ministry of Education and Research under the Music Discover project of the ACI-Masse de données

2. WEIGHTED MAXIMUM LIKELIHOOD PITCH ESTIMATOR

2.1. Main idea

This work focuses on signals which can be decomposed into a sum of sinusoidal components and a colored noise. In the following, a moving average process is assumed for the latter, with a corresponding FIR filter of transfer function $B(z)$. The spectral envelope of the partials is modeled by an autoregressive filter of transfer function $\frac{1}{A(z)}$. The technique presented herein is based on the decomposition of the set of DFT frequencies into two subsets: the subset $\mathcal{N}$ owing to the background noise properties and the other, $\mathcal{H}$, associated with the sinusoidal part. Once both $1/A(z)$ and $B(z)$ have been estimated, the constructed likelihood is maximized for the true value of F_0 since it simultaneously whitens the noise sub-spectrum and the sinusoidal sub-spectrum. In the case where a bad F_0 candidate is selected, the choice of a FIR $\mathcal{N}$-support sub-spectrum and an AR $\mathcal{H}$-support sub-spectrum ensures that such a flatness of both sub-spectra is not achieved.

2.2. Statistical framework

Let $\boldsymbol{x}$ denote the N-dimensional vector containing N successive samples of data, $\boldsymbol{X}$ the N-dimensional vector of its Digital Fourier Transform (DFT) and $\boldsymbol{F}$ the $N \times N$ orthonormal DFT matrix ($F_{(p,q)} = \frac{1}{\sqrt{N}} e^{-2i\pi \frac{pq}{N}}$). We assume that $\boldsymbol{x}$ results from the circular filtering of a centered white complex Gaussian random vector $\boldsymbol{w}$ of variance σ^2. Let $\boldsymbol{h}$ be the corresponding impulse response vector, and $\boldsymbol{H}$ its N-dimensional DFT vector. Since $\boldsymbol{X} = \mathrm{diag}\{\boldsymbol{H}\}\boldsymbol{F}\boldsymbol{w}$, $\boldsymbol{X}$ is a centered Gaussian random vector of covariance matrix $\sigma^2 \,\mathrm{diag}\{|\boldsymbol{H}|^2\}$.

Below, we consider that the observed data consist of a subset $\mathcal{S}$ of the DFT coefficients in vector $\boldsymbol{X}$. Then the previous discussion shows that the probability law of the

observed data is

$$p(X_{\mathcal{S}}) = \prod_{k\in\mathcal{S}} \frac{1}{\pi\sigma^2|H(k)|^2} e^{-\frac{|X(k)|^2}{\sigma^2|H(k)|^2}}.$$

Thus the normalized log-likelihood $L_{\mathcal{S}}(\sigma,\boldsymbol{h}) = \frac{1}{\#\mathcal{S}}\ln p(X_{\mathcal{S}})$ can be written in the form

$$L_{\mathcal{S}}(\sigma,\boldsymbol{h}) = C + \frac{1}{\#\mathcal{S}}\sum_{k\in\mathcal{S}}\left[\ln\left(\frac{|X(k)|^2}{\sigma^2|H(k)|^2}\right) - \frac{|X(k)|^2}{\sigma^2|H(k)|^2}\right] \quad (1)$$

where $C = -\frac{1}{\#\mathcal{S}}\sum_{k\in\mathcal{S}}\ln(\pi|X(k)|^2)$ is a constant with respect to σ and $\boldsymbol{h}$, and $\#\mathcal{S}$ denotes the number of elements in $\mathcal{S}$. Normalizing the likelihood by factor $1/\#\mathcal{S}$ aims at obtaining comparable, homogeneous values when $\#\mathcal{S}$ varies. Maximizing $L_{\mathcal{S}}$ with respect to σ yields the estimate

$$\widehat{\sigma}^2 = \frac{1}{\#\mathcal{S}}\sum_{k\in\mathcal{S}}\left|\frac{X(k)}{H(k)}\right|^2. \quad (2)$$

Then substituting equation (2) into equation (1) yields

$$L_{\mathcal{S}}(\boldsymbol{h}) \triangleq L_{\mathcal{S}}(\widehat{\sigma}^2,\boldsymbol{h}) = C - 1 + \ln\left(\rho_{\mathcal{S}}(\boldsymbol{h})\right) \quad (3)$$

where

$$\rho_{\mathcal{S}}(\boldsymbol{h}) = \frac{\left(\prod_{k\in\mathcal{S}}\left|\frac{X(k)}{H(k)}\right|^2\right)^{\frac{1}{\#\mathcal{S}}}}{\frac{1}{\#\mathcal{S}}\sum_{k\in\mathcal{S}}\left|\frac{X(k)}{H(k)}\right|^2} \quad (4)$$

is equal to the ratio between the geometrical mean and the arithmetical mean of the set $\left\{\left|\frac{X(k)}{H(k)}\right|^2\right\}_{k\in\mathcal{S}}$. Such a ratio is maximal and equal to 1 when $|X(k)/H(k)|$ is constant, independant of k, which means that $\rho_{\mathcal{S}}(\boldsymbol{h})$ measures the *whiteness*, or the *flatness* of $\left\{\frac{X(k)}{H(k)}\right\}_{k\in\mathcal{S}}$. The next step consists in choosing a parametric model for $\boldsymbol{h}$, and maximizing $L_{\mathcal{S}}$ with respect to the filter parameters. This optimization results in maximizing $\rho_{\mathcal{S}}(\boldsymbol{h})$. For instance, if $\boldsymbol{h}$ is modeled as an autoregressive (AR) filter, an approximate solution $\widehat{\boldsymbol{h}}$ to the optimization problem can be obtained by means of linear prediction techniques [9]. If $\boldsymbol{h}$ is modeled as a finite impulse response (FIR) filter of length $p \ll N$, an approximate solution $\widehat{\boldsymbol{h}}$ can be obtained by windowing a biased estimate of the autocovariance function.

2.3. Application to pitch estimation

Our pitch estimator relies on a weighted maximum likelihood (WML) method: for all subsets $\mathcal{H}$, *i.e.* for all possible F_0's, we calculate the weighted likelihood

$$L_{\mathcal{H}} = \alpha\ln\hat{\rho_{\mathcal{H}}} + (1-\alpha)\ln\hat{\rho_{\mathcal{N}}} \quad (5)$$
$$\text{with } \begin{cases} \hat{\rho_{\mathcal{H}}} = \max\limits_{A} \rho_{\mathcal{H}}\left(\frac{1}{A(z)}\right) \\ \hat{\rho_{\mathcal{N}}} = \max\limits_{B} \rho_{\mathcal{N}}\left(B(z)\right) \end{cases}$$

where $\mathcal{N} = \overline{\mathcal{H}}$ is the complement set of $\mathcal{H}$ and $0 < \alpha < 1$ (in practice we choose $\alpha = 1/2$). The pitch estimate is given by the set $\hat{\mathcal{H}}$ which maximizes $L_{\mathcal{H}}$. This maximum depends on the sum of the two $\mathcal{H}$-dependent terms in (5): $\ln\hat{\rho_{\mathcal{H}}}$ and $\ln\hat{\rho_{\mathcal{N}}}$. The flatness $\hat{\rho_{\mathcal{H}}}$ of the whitened components has a local maximum for a smooth spectral envelope, obtained when analyzing the true F_0 (see figure 1) or one of its multiples (*i.e.* $\mathcal{H}$ is a subset of the right set of overtones, see figure 2), or when $\mathcal{H}$ only contains noisy components. Low values of $\hat{\rho_{\mathcal{H}}}$ are obtained when amplitudes at the frequencies of $\hat{\mathcal{H}}$ are alternately low and high since AR filters have no zero, which means that they cannot fit a spectrum where some sinusoidal components in $\mathcal{H}$ are missing. This particularly happens for a sub-harmonic of the true F_0 (see figure 3). In other respects, when considering the spectral envelope of the noisy part of the sound, FIR filters have no pole, which means that they cannot fit any sinusoidal component: the spectral flatness $\hat{\rho_{\mathcal{N}}}$ of the whitened residual part reaches high values when the frequencies of overtones have been selected in $\mathcal{H}$, *i.e.* when analyzing any sub-harmonic frequency of the true F_0 (see figure 3). As illustrated in figure 4, by combining both spectral flatnesses $\hat{\rho_{\mathcal{H}}}$ and $\hat{\rho_{\mathcal{N}}}$, a global maximum is found for the true F_0 while any other local maximum in $\hat{\rho_{\mathcal{H}}}$ (or $\hat{\rho_{\mathcal{N}}}$) is attenuated by $\hat{\rho_{\mathcal{N}}}$ (or $\hat{\rho_{\mathcal{H}}}$), particularly harmonics and sub-harmonics.

3. APPLICATION TO MULTI-PITCH ESTIMATION OF PIANO TONES

3.1. Inharmonicity in piano tones

In a piano note, the stiffness of strings causes the frequencies of overtones to slightly differ from a perfect harmonic distribution. We are focussing one these quasi-harmonic sounds and exclude from this study other inharmonic tones like bell tones. The frequency of the overtone of order n is thus given by the inharmonicity law [10]:

$$f_n^{(f_0,\beta)} = nf_0\sqrt{1+\beta\left(n^2-1\right)} \quad (6)$$

where f_0 is the fundamental frequency and β is the inharmonicity coefficient. Note that β varies along the range of the piano keyboard and from one instrument to the other. Thus, the set $\mathcal{H}$, characterized by these two parameters, is defined as:

$$\mathcal{H}^{(f_0,\beta)} = \left\{f_n^{(f_0,\beta)}/n \in \mathbb{N}, f_n^{(f_0,\beta)} < F_s/2\right\} \quad (7)$$

Analysis of a synthetic signal with fundamental frequency 1076.6602 Hz.

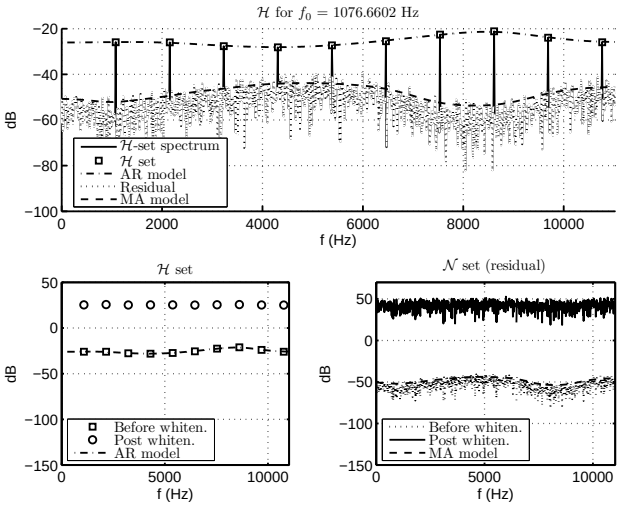


Figure 1: $L_{\mathcal{H}}$ estimation for $\mathcal{H} = \hat{\mathcal{H}}$ (true F_0). Overtones are selected in the spectrum (top), amplitudes of components fit the AR model (bottom left) and the residual spectrum is well whitened by the MA model (bottom right). In order to avoid overlapping between curves in the graphical representation, an constant offset is added to post-whitening dB-curves.

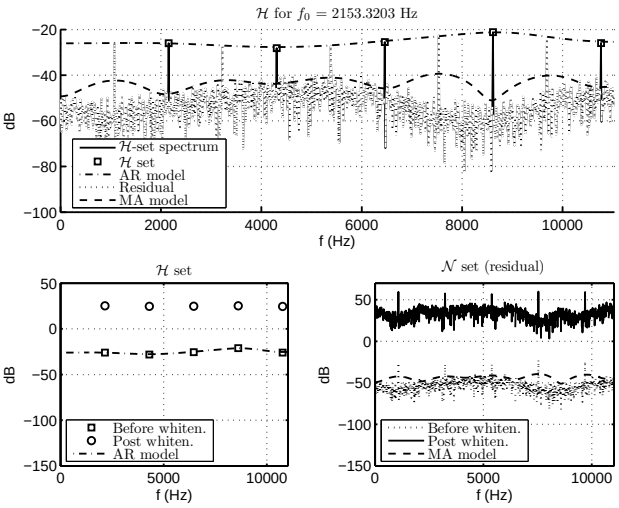


Figure 2: $L_{\mathcal{H}}$ estimation at twice the true F_0. Amplitudes of components fit the AR model whereas the residual spectrum is not perfectly whitened by the MA model, due to remaining components.

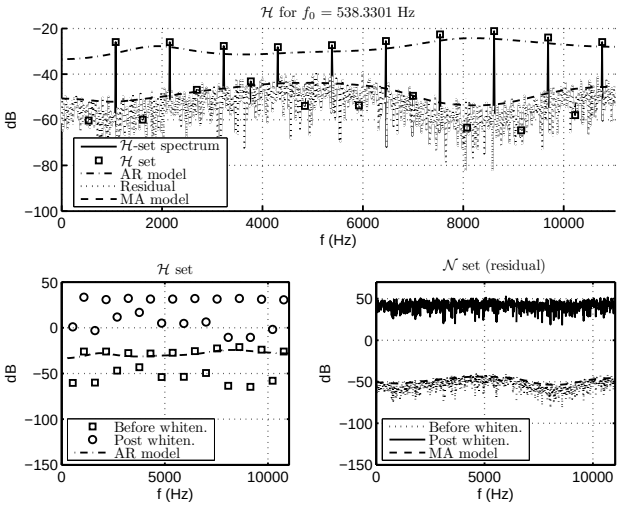


Figure 3: $L_{\mathcal{H}}$ estimation at half the true F_0. While residual spectrum is well whitened by the MA model, amplitudes of components do not fit the AR model, resulting in a low flatness of whitened amplitudes (bottom left, circles).

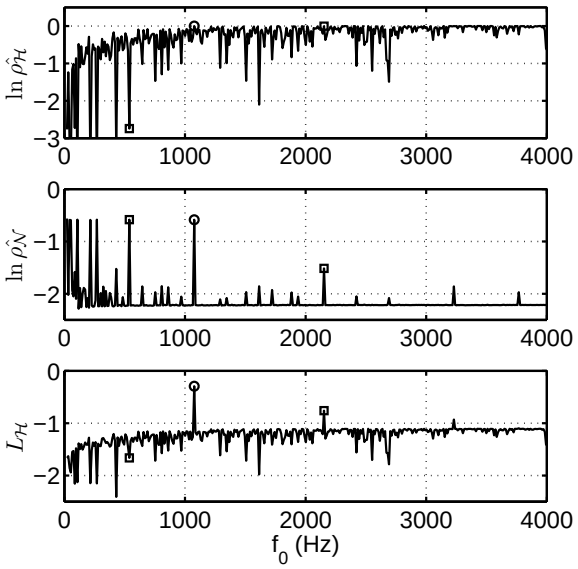


Figure 4: $\mathcal{H}$-dependent terms $\ln \hat{\rho_{\mathcal{H}}}$ (top) and $\ln \hat{\rho_{\mathcal{N}}}$ (middle), and weighted likelihood $L_{\mathcal{H}}$ (bottom), computed for all possible F_0's (*i.e.* all possible $\mathcal{H}$'s).

where F_s is the sampling frequency. Optimizing the log-likelihood $L\left(\mathcal{H}^{(f_0,\beta)}\right)$ with respect to $\mathcal{H}^{(f_0,\beta)}$ then consists in maximizing it with respect to f_0 and β.

3.2. From the theoretical model to real sounds

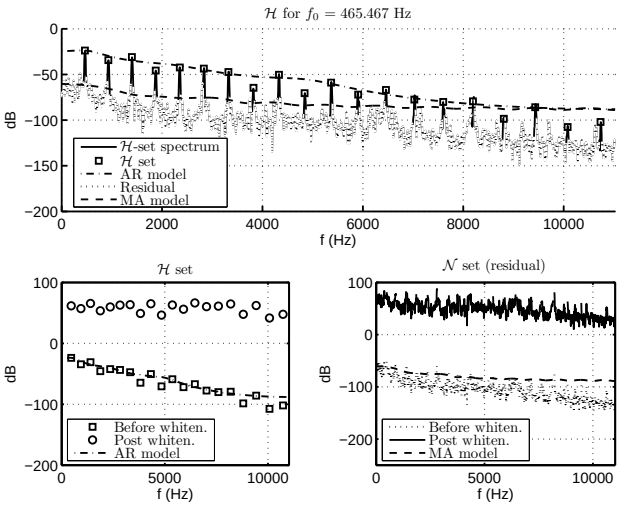


Figure 5: Real piano tone: separation between note components and residual part, and related MA and AR models

How do real piano tones fit the signal model described above? The AR model for the sinusoidal component, the MA noise model and the inharmonicity distribution of frequencies seem to be robust hypotheses. Conversely, the practical application of the method has to cope with two deviations from the theoretical point of view:

1. the assumption that f_n lies in the exact center of a frequency bin (multiple of $1/N$) is usually false, and spectral leakage thus influences the $\mathcal{N}$-support sub-spectrum.

2. the amplitude of the overtone may vary within the analysis frame, reflecting various effects as the energy loss of the sound and the beating between close adjacent components. This can affect the spectral envelope of the $\mathcal{H}$-support sub-spectrum.

The windowing of the analyzed waveform by a Hann window has proved to be a robust trade-off to overcome these issues. It prevents the spectral leakage associated with high energy components from masking weak overtones. Amplitudes of every overtone k are estimated by performing a parabolic interpolation of the spectrum (in decibels) based on the values in the nearest Fourier bins. The resulting (linear) value is used when computing the sinusoidal-part spectral flatness $\hat{\rho_{\mathcal{H}}}$, *i.e.* in place of $X(k)$ in equation (4). In order to minimize the effects described above in $\hat{\rho_{\mathcal{N}}}$ (see equation (4)), primary lobes of the frequencies selected in $\mathcal{H}$ are removed from $\mathcal{N}$, which is redefined as:

$$\mathcal{N} = \{k'/\forall f \in \mathcal{H}, |k'/N - f| > \Delta f/2\} \quad (8)$$

where Δf is the width of the primary lobe ($\Delta f = \frac{4}{N}$ for a Hann window). Note that the question of removing a set of components is a key step in the implementation of our algorithm. As shown in figure 5, the proposed method performs an approximate removal that offers a satisfying trade-off between efficiency and computational cost. Other techniques based on amplitude estimation and adapted filter design have been tested without bringing major improvements. The non-stationary nature of signals seems to be responsible for this limitation. It should be taken into account for enhancing the separation between a set of components and the residual signal.

3.3. Extension to polyphonic sounds

We now consider that the deterministic signal $s(n)$ is a sum of M inharmonic sounds: $s(n) = \sum_{m=1}^{M} s^{(m)}(n)$ and $\forall m \in \{1 \dots M\}$, $f_n^{(m)} = n f_0^{(m)} \sqrt{1 + \beta^{(m)}(n^2 - 1)}$, where $f_0^{(m)}$ is the pitch and $\beta^{(m)} > 0$ is the inharmonicity coefficient of the m^{th} tone. Each note is associated with one individual AR model, and weights in the likelihood are uniformly distributed among notes. Thus the WML principle consists in maximizing the log-likelihood:

$$L(\mathcal{H}^{(1)}, \dots, \mathcal{H}^{(M)}) = \frac{1}{2M} \sum_{m=1}^{M} \ln \rho_{\mathcal{H}^{(m)}} \left(\frac{1}{A^{(m)}(z)} \right) + \frac{1}{2} \ln \hat{\rho_{\mathcal{N}}} \quad (9)$$

where $\mathcal{H}^{(m)} = \mathcal{H}^{\left(f_0^{(m)}, \beta^{(m)}\right)}$ and $\mathcal{N}$ is the set of bins outside primary lobes of frequencies of any $\mathcal{H}^{(m)}$. The optimization is performed with respect to each of the sets $\mathcal{H}^{(1)}, \dots, \mathcal{H}^{(M)}$. Each set $\mathcal{H}^{(m)}$ is defined by the parameters $\{(f_0^{(m)}, \beta^{(m)})\}_{m \in \{1 \dots M\}}$ and $1/A^{(m)}(z)$ is the AR filter related to note m. Two distinct sets $\mathcal{H}^{(m_1)}$ and $\mathcal{H}^{(m_2)}$ may intersect, allowing overlap between spectra of notes m_1 and m_2. The algorithm presented in section 2.3 can be applied straightforwardly.

3.4. Multi-pitch estimator implementation

Multi-pitch estimation is often performed either in an iterative or in a joint process. The proposed method belongs to the joint estimation category. While iterative methods consist in successively estimating and removing a predominant F_0, joint estimation simultaneously extracts the set of

F_0's. Thus, a direct implementation of the algorithm described above would require to compute the ML of all possible combinations of notes, leading to a high-order combinatory task. For instance, more than 2.10^6 different chords exist for a 4-note polyphony in the full piano range, each of these candidates requiring several calls to the likelihood function since the exact F_0 and β values are unknown.

In order to reduce the cost of the ML estimation, a two-step algorithm is proposed. First, each possible chord is evaluated on a reduced number of points N_p in the $(f_0^{(m)}, \beta^{(m)})$ region around F_0 values from the well-tempered scale and approximate β values. N_{cand} chord candidates are extracted among all combinations by selecting the N_{cand} greatest likelihood values. Then, the likelihood of each selected candidate is locally maximized with respect to coefficients $f_0^{(m)}$ and $\beta^{(m)}$. A simplex method is used to perform this optimization, which is initialized with the $f_0^{(m)}$ and $\beta^{(m)}$ values selected during the first step. Finally, the chord with maximum accurately-computed likelihood is selected as the chord estimate.

4. EXPERIMENTAL RESULTS

The algorithm has been tested on a database composed of about 540 isolated piano tones of the RWC database [11] and random chords generated by several virtual piano softwares based on sampled sounds. About 600 two-note chords and 600 three-note chords were evaluated. In each case, the polyphony is known a priori by the algorithm and the estimation results from the analysis of one 93 ms frame, beginning 10 ms after the onset. F_0 estimates are rounded to the nearest half-tone in the well-tempered scale in order to determine if an estimated note is correct. This approximation on F_0 is carried out in order to evaluate the pitch estimation at a note level rather than at a frequency level. The note search range spreads over 5 octaves, from MIDI note 36 ($f_0 = 65$ Hz) to MIDI note 95 ($f_0 = 1976$ Hz). These test conditions are similar to the ones used in competitor systems [4, 5, 7] in terms of frame length, F_0 search range and error rate definition.

The parameters of the system have been adjusted as follows. Sounds are sampled at 22050 Hz. DFT are computed on 4096 points after zero-padding the 2048-point frame. The AR model order is set to 8, the MA model order to 20. In the first step of the implementation described in section 3.4, all chord combinations are evaluated, each one with $N_p = 10$ (polyphony ≤ 2) or $N_p = 5$ (polyphony three) different $(f_0^{(m)}, \beta^{(m)})$ values. Then $N_{cand} = 75$ (monophony) or $N_{cand} = 150$ (polyphony ≥ 2) chord candidates are selected for the second step.

Error rates are 2.0% in monophony, 7.5% in polyphony two and 23.9% in polyphony three. They are reported in

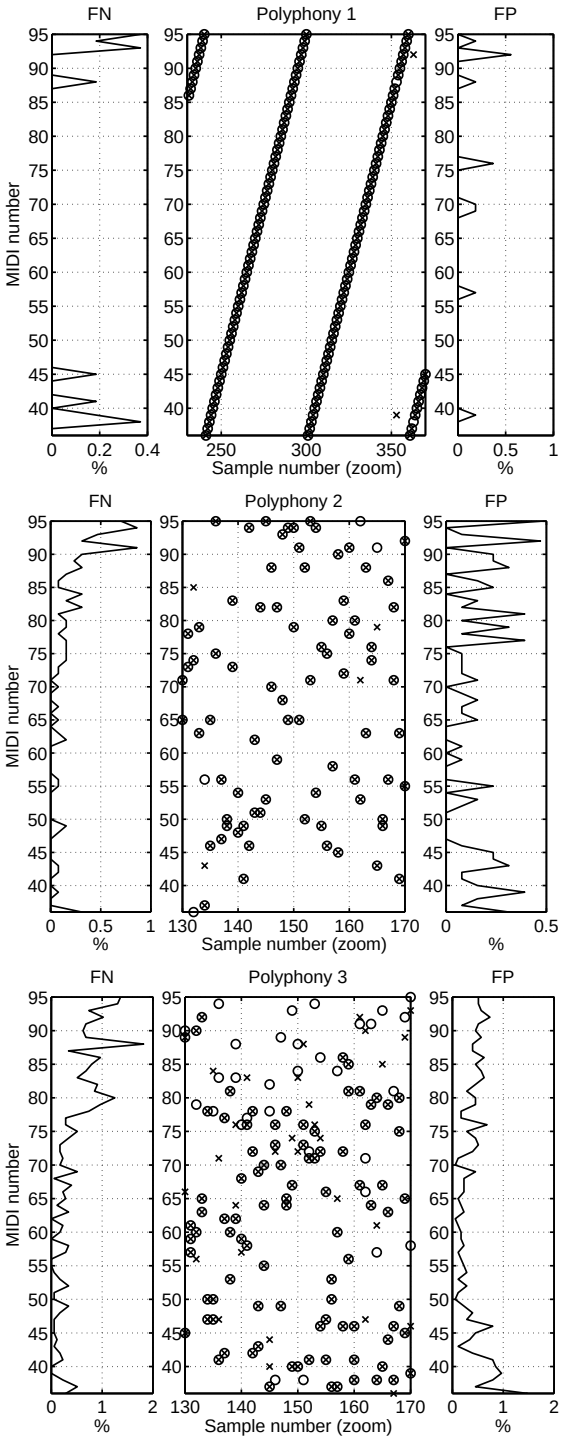


Figure 6: Estimation results: for a given polyphony (1 to 3 from top to bottom), random chords are generated (circles) and estimated (crosses). For visual representation clarity, only 50 samples of them are shown (center). Distribution of false negatives is displayed on the left. Distribution of false positives is displayed on the right.

Polyphony	1	2	3
Error rate	2.0% $\pm 0.6\%$	7.5% $\pm 0.75\%$	23.9% $\pm 1.0\%$
Octave error rate	0%	1.6%	5.2%
State of the art	$2 \sim 11\%$	$7 \sim 25\%$	$\approx 10 \sim 35\%$

Table 1: Error rates with respect to polyphony. Lower and upper bounds of state-of-the-art performances are also reported. Confidence interval is derived as the standard deviation of the error rate estimator.

table 1 and can be compared to the three competitor systems previously mentionned. Their performances have been established in [7] for polyphony one, two, four and six: error rates vary from 2 to 11% in monophony, from 7 to 25% in polyphony two and from 14 to 41% in polyphony four. Error rates in polyphony three are not given, but could be figured out as intermediate values between results in polyphony two and four, which would lead to approximate error rates between 10 and 35%. The proposed pitch estimator is comparable to competitor systems in terms of performance. Error rates are particularly competitive in polyphonies one and two.

The evaluation task has been performed using randomly uniformly-distributed notes in order to provide experimental results from an objective point of view rather than from musical considerations. The distribution of errors is reported in figure 6. The few errors in polyphony one occur in the lowest and highest pitch regions. In polyphony two and three, most of missed notes (or false negatives, FN) are located in the treble part of the piano range whereas the false-alarm notes (or false positives, FP) estimated in place of them tend to be distributed in a more uniform manner along the piano range. Closely-spaced chords in the medium range seem easier to detect than widely-spaced chords. Octave error are scarce – around one fifth of all errors for each polyphony number –, which can be explained by the complementary contributions of note and noise likelihoods. On the contrary, high-pitched FN and large-interval errors often occur, in spite of the likelihood normalization stage, due to the sensitivity of the ML approach to the variable number of frequency parameters that depends on F_0 candidates.

5. CONCLUSIONS

The multipitch estimation task has been performed here through a Maximum Likelihood approach. It consists in modeling notes and residual noise by AR and MA models, and results in a criterion on their spectral flatness after a whitening process based on the models. The method has been validated by satisfying experimental results for polyphony one to three.

Future works will deal with managing the overlap between notes spectra, with improving the model for the spectral envelope of notes and with making the computational cost decrease in order to both benefit from the efficiency of the estimator and avoid the inherent complexity of joint estimation of multiple F_0's.

6. REFERENCES

[1] A. de Cheveigne and H. Kawahara, "YIN, a fundamental frequency estimator for speech and music," *JASA*, vol. 111, no. 4, pp. 1917–1930, 2002.

[2] M. Ryynänen and A.P. Klapuri, "Polyphonic music transcription using note event modeling," in *Proc. of WASPAA*, New Paltz, NY, USA, October 2005, IEEE, pp. 319–322.

[3] M. Marolt, "A connectionist approach to automatic transcription of polyphonic piano music," *IEEE Trans. on Multimedia*, vol. 6, no. 3, pp. 439–449, 2004.

[4] T. Tolonen and M. Karjalainen, "A computationally efficient multipitch analysis model," *IEEE Trans. on Speech and Audio Processing*, vol. 8, no. 6, pp. 708–716, 2000.

[5] A.P. Klapuri, "Multiple fundamental frequency estimation based on harmonicity and spectral smoothness," *IEEE Trans. on Speech and Audio Processing*, vol. 11, no. 6, pp. 804–816, November 2003.

[6] G. Peeters, "Music pitch representation by periodicity measures based on combined temporal and spectral representations," in *Proc. of ICASSP 2006*, Toulouse, France, May 14-29 2006, IEEE, vol. 5, pp. 53–56.

[7] A.P. Klapuri, "A perceptually motivated multiple-f0 estimation method," in *Proc. of WASPAA*, New Paltz, NY, USA, October 2005, IEEE, pp. 291– 294.

[8] Manuel Davy, Simon Godsill, and Jerome Idier, "Bayesian analysis of polyphonic western tonal music," *JASA*, vol. 119, no. 4, pp. 2498–2517, 2006.

[9] S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1993.

[10] N. H. Fletcher and T. D. Rossing, *The Physics of Musical Instruments*, Springer, 1998.

[11] T. Nishimura M. Goto, H. Hashiguchi and R. Oka, "RWC music database: Music genre database and musical instrument sound database," in *Proc. of ISMIR*, Baltimore, Maryland, USA, 2003, pp. 229–230.

EFFICIENT DESCRIPTION AND RENDERING OF COMPLEX INTERACTIVE ACOUSTIC SCENES

Jean-Marc Jot

Creative Advanced Technology center
Scotts Valley, USA
jmj@atc.creative.com

ABSTRACT

Interactive environmental audio spatialization technology has become commonplace in personal computers and is migrating into portable entertainment platforms (including cell phones) and multi-player game servers (virtual online worlds). While the primary current application of this technology is 3D game sound track rendering, it is ultimately necessary in the implementation of any personal or shared immersive virtual world ("virtual reality"). The successful development and deployment of such applications in new mobile or online platforms involves maximizing the plausibility of the synthetic 3D audio scene while minimizing the computational and memory footprint of the audio rendering engine. It also requires a flexible, standardized scene description model to facilate the development of applications targeting multiple platforms. This paper reviews a computationally efficient 3-D positional audio and spatial reverberation processing architecture for real-time virtual acoustics over headphones or loudspeakers, compatible with current interactive audio standards (including MPEG-4, OpenAL, JSR 234 and OpenSL ES).

1. INTRODUCTION AND OVERVIEW

The applications of interactive 3D audio technologies include simulation and training, telecommunications, video games, multimedia installations, movie or video soundtracks, and computer music [1]-[5].

Virtual acoustics technology has its origins in research carried out in the 1970's, which targeted two distinct applications:
- *Architectural acoustics:* Schroeder et al. developed simulation methods based on geometrical acoustics to derive a computed echogram from a physical model of room boundaries and the source and listener positions [6];
- *Computer music:* Chowning developed a 4-channel spatialization system for simulating dynamic movements of sounds, which provided direct control of two perceptual control parameters for each source: apparent direction of sound arrival and apparent distance to the listener, along with a derived Doppler shift [7]. Artificial reverberation was included to enhance the robustness of distance effects. Later, Moore proposed an extension of this approach where early reflections were controlled indirectly via a geometrical acoustic model [8].

Interactive virtual acoustics systems require real-time rendering and mixing of multiple audio streams (sound sources) to feed a set of loudspeakers or headphones. This rendering system is driven by an acoustic scene description model which provides positional and environmental audio parameters for all sound sources.

The scene description represents a virtual world including sound sources and one or more listeners within an acoustical environment which may incorporate one or more rooms and acoustic obstacles.

Standardization is essential for enabling platform-independent playback and re-usability of scene elements by application authors and sound designers. Current standard interactive audio scene description models include high-level scripting languages such as the MPEG-4 Advanced Audio Binary Format for Scene description (AABIFS) [9] and low-level application programming interfaces used in the creation of video games, such as OpenAL, JSR 234 and OpenSL ES [10]-[12]. In this paper, we will consider a generic, low-level scene description model based on OpenAL [10] and its environmental extensions, I3DL2 [13] and EAX [14]-[15]. For applications that require higher-level world representations, a real-time translation software layer can be implemented above the rendering engine to convert the high-level representation to low-level description parameters [14].

In the first section of this paper, we discuss and compare digital signal processing methods for computationally efficient real-time spatialization of multiple sound sources over headphones or loudspeakers. This includes discrete amplitude panning, Ambisonic and binaural or transaural techniques [16]-[25] and introduces a recently developed multi-channel binaural synthesis method based on discrete spatial functions, previously introduced in [26]. The description model and rendering methods are then extended to include the acoustic effects of the listener's immediate environment. This includes the effects of acoustic obstacles and room boundaries or partitions on the perception of each sound source. Acoustic reflections and room reverberation are rendered by use of feedback delay networks [27]-[30]. A statistical reverberation model, previously introduced in [30], is included for modelling per-source distance and directivity effects. We further extend the model to account for the presence of acoustic environments or rooms adjacent to the listener's environment. An efficient spatial reverberation and mixing architecture, previously introduced in [26], is described for the spatialization of multiple sound sources around a virtual listener navigating across multiple connected virtual rooms. This processing architecture includes a novel cost-efficient method for simulating multiple spatially extended sound sources or sound events.

The models and methods reviewed in this paper enable the realization of comprehensive, computationally efficient, flexible and scalable high-quality interactive 3D audio rendering systems for deployment in a variety of consumer appliances (ranging from personal computers to home theater and mobile entertainment systems) and services (including multi-user comunication and telepresence).

2. REFERENCES

[1] D. R. Begault, *3-D Sound for Virtual Reality and Multimedia* (Academic Press, New York, 1994).

[2] M. Kleiner, B.-I. Dalenbäck, and P. Svensson, "Auralization - an Overview," *J. Audio Eng. Soc.* 41(11): 861-875 (1993 Nov.).

[3] M. Cohen and E. Wenzel, E, "The Design of Multidimensional Sound Interfaces," Tech. Rep. 95-1-004, Human Interface Laboratory, Univ. of Aizu (1995).

[4] J.-M. Jot, "Real-time Spatial processing of sounds for music, multimedia and interactive human-computer interfaces," *ACM Multimedia Systems J.* 7(1) (1999 Jan.).

[5] A. Harma & al., "Augmented Reality Audio for Mobile and Wearable Appliances," *J. Audio Eng. Soc.* 52(6): 618-639 (2004 June).

[6] M. R. Schroeder, "Computer Models for Concert Hall Acoustics," *American J. Physics* 41:461-471 (1973).

[7] J. Chowning, "The Simulation of Moving Sound Sources.," *J. Audio Eng. Soc.* 19(1) (1971).

[8] F. R. Moore, "A General Model For Spatial Processing of Sounds," *Computer Music J.* 7(6) (1983)

[9] R. Väänänen and J. Huopaniemi, "Advanced AudioBIFS: Virtual Acoustics Modeling in MPEG-4 Scene Description," *IEEE Trans. Multimedia* 6(5): 661-675 (2004 Oct.)

[10] G. Hiebert & al., "OpenAL 1.1 Specification and Reference," www.openal.org (1995 June)

[11] M. Paavola & al., "JSR 234: Advanced Multimedia Supplements," Java Community Process spec. www.jcp.org (2005 June).

[12] Khronos Group, "OpenSL ES – Open Standard Audio API for Embedded Systems", www.khronos.org/opensles (2007).

[13] J.-M. Jot & al., "IA-SIG 3D Audio Rendering Guideline, Level 2" (I3DL2) www.iasig.org (1999).

[14] J.-M. Trivi and J.-M. Jot, "Rendering MPEG-4 AABIFS Content Through a Low-level Cross-platform API," Proc Int. Conf. Multimedia (ICME 2002).

[15] J.-M. Jot and J-.M. Trivi, "Scene Description Model and Rendering Engine for Interactive Virtual Acoustics," Proc. 120th Conv. Audio Eng. Soc., preprint 6660 (2006 May).

[16] V. Pulkki, "Virtual Sound Source Positioning Using Vector Base Amplitude Panning," *J. Audio Eng. Soc.* 45(6): 456-466 (1997 June).

[17] M. A. Gerzon, "General Metatheory of Auditory Localization," Proc. 92nd Conv. Audio Eng. Soc., preprint 3306 (1992).

[18] M. A. Gerzon, "Ambisonics in Multichannel Broadcasting and Video," *J. Audio Eng. Soc.* 33(11) (1985).

[19] D. G. Malham and A. Myatt, "3-D Sound Spatialization Using Ambisonic Techniques," *Computer Music J.* 19(4) (1995).

[20] D. H. Cooper and J. L. Bauck, "Prospects for Transaural Recording," *J. Audio Eng. Soc.* 37(1/2) (1989).

[21] J.-M. Jot, V. Larcher, and O. Warusfel, "Digital Signal Processing Issues in the Context of Binaural and Transaural Stereophony," Proc. 98th Conv. Audio Eng. Soc., preprint 3980 (1995).

[22] W. G. Gardner, *3-D Audio Using Loudspeakers*, Ph. D. Thesis, Massachussets Institute of Technology (1997),

[23] A. Jost and J.-M. Jot "Transaural 3-D Audio with User-controlled Calibration," *Proc. Int. Conf on Digital Audio Effects* (DAFX 2000).

[24] J.-M. Jot, V. Larcher, and J.-M. Pernaux, "A Comparative Study of 3-D Audio Encoding and Rendering Techniques," *Proc. 16th Int. Conf. Audio Eng. Soc.* (1999 March).

[25] V. Larcher, J.-M. Jot, G. Guyard, and O. Warusfel, "Study and Comparison of Efficient Methods for 3-D Audio Spatialization Based on Linear Decomposition of HRTF Data", Proc. 108th Conv. Audio Eng. Soc., preprint 5097 (2000 Jan.).

[26] J.-M. Jot, M. Walsh and A. Philp, "Binaural Simulation of Complex Acoustic Scenes for Interactive Audio," Proc. 121st Conv. Audio Eng. Soc., preprint 6950 (2006 Oct.).

[27] J.-M. Jot, "Efficient Models for Reverberation and Distance Rendering in Computer Music and Virtual Audio Reality", *Proc. International Computer Music Conference* (1997).

[28] W. G. Gardner, "Reverberation algorithms," *Applications of Signal Processing to Audio and Acoustics* (ed. M. Kahrs, K. Brandenburg), Kluwer Academic (1998).

[29] L. Dahl and J.-M. Jot, "A Reverberator Based on Absorbent All-pass Filters", *Proc. Int. Conf on Digital Audio Effects* (DAFX 2000).

[30] J.-M. Jot, L. Cerveau, and O. Warusfel, "Analysis and Synthesis of Room Reverberation Based on a Statistical Time-Frequency Model," Proc. 103rd Conv. Audio Eng. Soc., preprint 4629 (1997 Aug.).

2ND ORDER SPHERICAL HARMONIC SPATIAL ENCODING OF DIGITAL WAVEGUIDE MESH ROOM ACOUSTIC MODELS

Alex Southern

Audio Lab - Intelligent Systems Group
Department of Electronics
University of York
York, UK
YO10 5DD
aps502@ohm.york.ac.uk

Damian Murphy

Audio Lab - Intelligent Systems Group
Department of Electronics
University of York
York, UK
YO10 5DD
dtm3@ohm.york.ac.uk

ABSTRACT

The aim of this research is to provide a solution for listening to the acoustics of Digital Waveguide Mesh (DWM) modelled virtual acoustic spaces. The DWM is a numerical simulation technique that has shown to be appropriate for modelling the propogation of sound through air. Recent work has explored methods for spatially capturing a soundfield within a virtual acoustic space using spatially distributed receivers based on sound intensity probe theory. This technique is now extended to facilitate spatial encoding using second-order spherical harmonics. This is achieved through an array of pressure sensitive receivers arranged around a central reference point, with appropriate processing applied to obtain the second-order harmonic signals associated with Ambisonic encoding/decoding. The processed signals are tested using novel techniques in order to objectively assess their integrity for reproducing a faithful impression of the virtual soundfield over a multi-channel sound system.

1. INTRODUCTION

By providing a solution for the auralization of physically modelled enclosed acoustic spaces, this research aims to provide a means of synthesising both spatially and psychoacoustically realistic room impulse responses (RIR). In the longer term the Digital Waveguide Mesh (DWM) modelled RIR's may help measure some of the well documented limits of the DWM and provide information about any resulting psychoacoustical artefacts. Other common approaches of room acoustics modelling include the image-source method [1] and ray-tracing as used by ODEON [2] and pyramid tracing as in RAMSETE [3]. This research particularly focuses on providing spatial auralization of the modelled acoustic space, ideally in a manner that reproduces a pyschoacoustically faithful soundfield at the listener's ears. Previous work has explored the capture and decoding of a DWM simulated virtual 2D soundfield [4]. An orthogonal arrangement of 5 omni-directional receivers (7 for 3D), called a crux of receivers, has been shown to be a suitable technique for capturing a first-order B-Format Room Impulse Response (RIR) in a DWM. This paper proposes to extend this method by encoding the DWM synthesised soundfield to audio channels corresponding to the second-order channels associated with the Ambisonic multi-channel system.

The remainder of this paper is arranged as follows. In section 2 the digital waveguide is introduced and its role discussed in forming a DWM used to spatially measure RIRs. Section 3 reviews Ambisonics and B-Format theory, both briefly in terms of the encoding and decoding/rendering stage. In section 4 there is a review of previous techniques and results followed by an explanation of the proposed encoding techniques to be applied to the DWM model. Section 5 details the various testing techniques used to assess the success of the process from different perspectives, with results presented and analysed in section 6. Finally section 7 concludes the paper and suggests future points for consideration.

2. THE DIGITAL WAVEGUIDE

Digital waveguide synthesis was first introduced by J.O. Smith as a 1D computational physical modelling technique which found application in musical instrument synthesis, speech synthesis and acoustics [5]. Briefly the theory states that a travelling wave through a 1D system may be considered as the summation of two waves travelling in opposite directions. Mathematically this is expressed using d'Alembert's solution to the 1D wave equation which may be used to model a tube or vibrating string.

$$y(t,x) = y_r\left(t - \frac{x}{c}\right) + y_l\left(t + \frac{x}{c}\right) \tag{1}$$

Where c is the speed of sound, t is the current point in time (seconds), x is the position along the string (metres) and y_r & y_l are the right and left going waves. The digital waveguide is a discrete implementation of (1) that in practice is represented using a bi-directional digital delay line as shown in Figure 1.

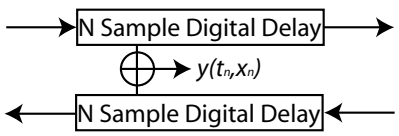


Figure 1: *The bi-directional digital delay line that forms the basic 1D digital waveguide.*

The discrete time implementation of (1) is given by (2) which mathematically expresses the use of the bi-drectional delay lines in Figure 1 forming the basic digital waveguide [6].

$$y(t_n, x_m) = y^+(n-m) + y^-(n+m) \tag{2}$$

Where $y(t_n, x_m)$ is the physical amplitude output at time sample t_n at position x_m along the bi-directional delay line and $n = nT$ and $m = mT$ when T is sampling interval in seconds. y^+ and y^- are digital equivalent of the continuous time domain signals y_r and y_l.

2.1. The Digital Waveguide Mesh

The DWM is comprised of multiple digital waveguides which may extend the system to 2D, 3D or higher dimensions. The extension of the digital waveguide to the 2D case has previously been discussed as a means of modelling membranes, e.g. [7]. In Figure 2 it may be observed that the nodes are connected by the same bi-directional delay lines shown in Figure 1. The delay between each node is 1 sample and it has been shown that these systems are capable of modelling a range of resonant objects with different degrees of dimensionality [8].

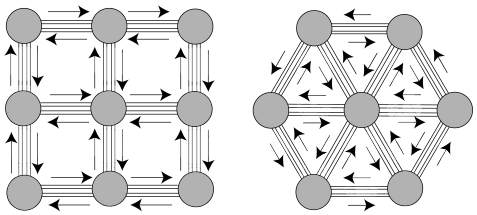


Figure 2: *The rectilinear and triangular DWM topologies*

This work has been conducted using a research tool called *RoomWeaver* which is a result of previous work at the University of York [9]. This software may be used to design virtual geometric representations of acoustic environments and then properly calculate the associated RIR. Recent work has shown how to synthesise B-Format RIRs from the DWM model in *RoomWeaver* [4] [10].

3. AMBISONICS ENCODING AND DECODING

Ambisonics describes a method of capturing/encoding a 3D soundfield by means of its spherical harmonic decompostion to the n^{th} order around a spatial reference point [11]. These signals may then be decoded to a loudspeaker arrangment, although the exact decoding equations will change primarily depending on the loudspeaker positions and many decoding schemes have been discussed previously e.g. [12] [13] [14]. Figure 3 shows the 3D polar responses associated with each channel of a 2nd order system. W is an omnidirectional pressure signal, and XYZ correspond to 3 velocity signals arranged orthogonally along the cartesian axes. Collectively these 1st order system signals WXYZ are known as B-Format. A full 2nd order system consists of the 1st order WXYZ channels and the channels RSTUV. The polar responses shown in Figure 3 are described in (3) using the appropriate corresponding ambisonic encoding equations [15].

$$
\begin{aligned}
W &= 0.707107 \\
X &= \cos(\theta_A) \cdot \cos(\theta_E) \\
Y &= \sin(\theta_A) \cdot \cos(\theta_E) \\
Z &= \sin(\theta_E) \\
R &= 1.5 \sin^2(\theta_E) - 0.5 \\
S &= \cos(\theta_A) \cdot \sin(2\theta_E) \\
T &= \sin(\theta_A) \cdot \sin(2\theta_E) \\
U &= \cos(2\theta_A) \cdot \cos^2(\theta_E) \\
V &= \sin(2\theta_A) \cdot \cos^2(\theta_E)
\end{aligned} \tag{3}
$$

Where θ_A and θ_E are the sound source azimuth and elevation respectively. Note that it is also possible to consider ambisonics for the horizontal plane only. In this case all expressions containing θ_E may be omitted as they all become equal to one when $\theta_E = 0$. Therefore the horizontal 1st order case only requires WXY channels. WXYUV channels are required for 2nd order horizontal ambisonics and therefore these are the channels that will be encoded from the DWM in this work.

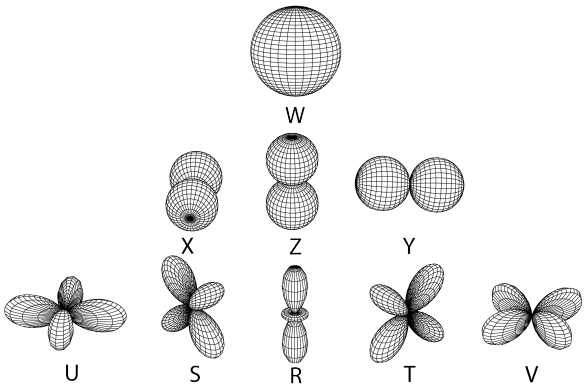


Figure 3: *Spherical harmonics upto 2nd order*

4. ENCODING TECHNIQUES

Two different approaches for encoding the DWM into B-Format have been considered previously. The first consisted of a circle of receivers with the second using an arrangement based on a crux of receivers, both of which are shown in Figure 4. Receivers are defined as pressure sensors that form the fundamental sampling method in the DWM. In [10] it was shown that the circle of receivers method does not perform reliably, primarily due to fact that it does not discrimate between wavefronts entering and leaving the circle.

Only the crux method will be considered here as this research focuses on extending the current technique to capture the 2nd order spherical harmonics. However a more comprehensive measurement method using a circle of 1st order receivers has been presented previously for real rooms in [16].

Figure 4: *The circle and 2D crux of receivers*

4.1. Capturing 1st Order Components

Previous work has shown that the 1st order spherical harmonics or B-Format signal may be obtained from the DWM [4] [10]. The fundamental technique on which this is based is a process more commonly associated with p-p sound intensity probes for the calculation of velocity at a point. Equation 4 expresses the velocity component in terms of two closely spaced pressure sensors [17].

$$u(t) \approx \left(\frac{1}{\rho_0 \cdot d}\right) \int_0^{\infty} \left[p_1(t) - p_2(t)\right] \delta t \tag{4}$$

Where ρ_0 is the density of air, d is the distance between pressure sensors and $p_1(t)$ and $p_2(t)$ are the pressure at time t at each sensor. For a more detailed account of using this technique for capturing the X and Y velocity components see [10]. However we will briefly include a preliminary study in order to demonstrate this technique with a view to extending the test to the 2nd order case. The test involves placing two closely spaced receivers into a simulated acoustic free field and then rotating the position of an impulse sound source at a fixed radius of 1 metre every 22.5°. Figure 5(a) illustrates the setup for the test. At each source position the two receivers are processed appropriately according to (4) and the sample value of the main peak is plotted against its angle with the results shown in Figure 5(b).

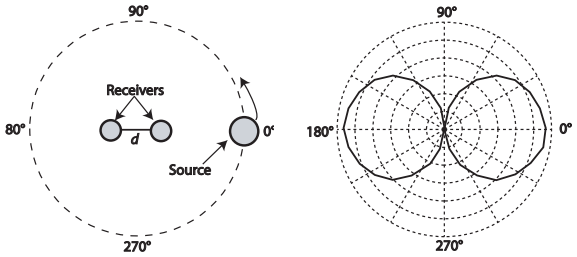


Figure 5: *Preliminary test for capturing ambisonic velocity components*

Therefore Figure 5(b) clearly illustrates that (4) provides directional discrimination which results in the corresponding velocity pattern.

4.2. Capturing 2nd Order Components

Horizontal 2nd order ambisonics comprises signals WXYUV. The method for obtaining X and Y has been described above, while W is simply captured by placing a receiver at the central reference position. The U and V channels are therefore required to complete the signal, a method is proposed in the following.

Prior to this paper, the TKK microphone arrangement has been shown to be capable of forming highly directional beams from a spaced array of omnidirectional capsules [18]. The microphone is described as being suitable for capturing 1st order B-Format signals [19]. For capturing 2nd order components it is presented that the microphone array is able to form the associated polar pickup patterns but for a reduced frequency band [18] [19]. More generally the process of using lower order microphone capsules to provide higher order directional responses has been termed the *Blumlein Difference Technique* and various cases are discussed in [20] [21].

Equations (5) and (6) describe U and V respectively in terms of the X and Y signals according to [18].

$$U = X^2 - Y^2 \tag{5}$$

$$V = X \cdot Y \tag{6}$$

Therefore in order to obtain the X and Y velocity components the arrangment of receivers shown in Figure 6(a) is used. Capturing the 1st order pressure gradient from two pressure sensors was discussed in section 4.1, while capturing the 2nd order pressure gradient may be achieved from (5) and (6).

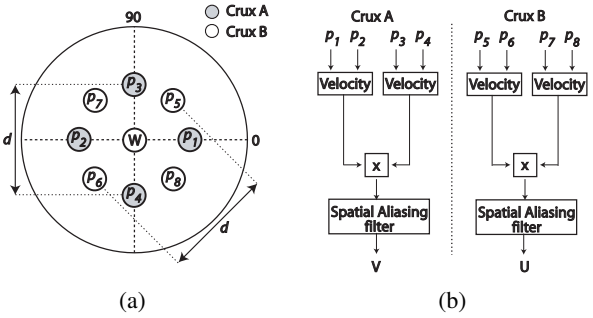
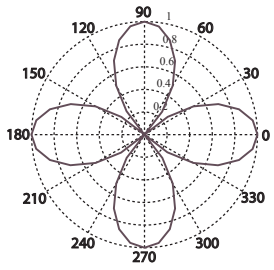


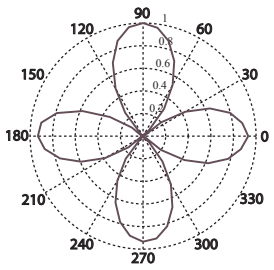
Figure 6: *Proposed arrangment of receivers for 2nd order encoding along with a basic encoding strategy*

Figures 6(a) & (b) break down the calculation of the UV channels from the 4 receivers labelled Crux A and Crux B. An alternative yet similar approach for processing the 2nd order components may be employed by extending the technique presented in [20] to obtain the receiver arrangement in Figure 6(a). Another preliminary test is presented which aims to initially verify the worth of the proposed 2nd order encoding technique. The test is carried out in the same manner as shown in Figure 5(a) however the difference being that the four receivers labeled Crux B are used as arranged in Figure 6(a). The processing carried out is used to obtain the polar pattern associated with the U channel, which is given in (3) and plotted in Figure 7(a). Figure 7(b) is the resulting polar pickup pattern of the array in Figure 6 after processing. For clarity Figure 7(c) is provided so that a direct comparison of the desired theoretical response and actual response can be made after normalising the amplitude values. Equations (5) & (6) may describe how to obtain the UV channels from using Crux A, however (5) requires two multiplies with the associated potential for a loss of accuracy, hence resulting in greater numerical error. Consequently Crux B is used to minimize this effect.

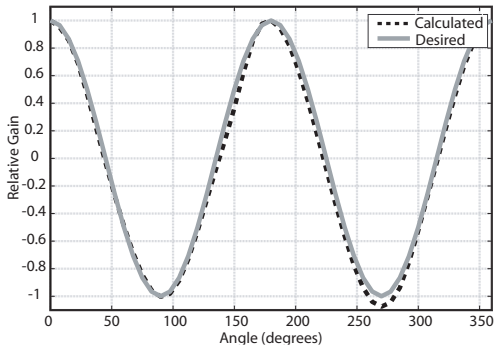
This preliminary test has confirmed that it is possible to obtain the necessary gain response for the U channel using the X and Y velocity signals. However the test tells us nothing about the directionally dependent frequency response or the reliability of this technique for facilitating higher resolution spatial rendering of virtual acoustic spaces.



(a) Theoretical Response



(b) Preliminary test response



(c) Theoretical and test responses superimposed

Figure 7: *Preliminary test results for 2nd order encoding*

5. TESTING TECHNIQUES

Numerous testing strategies are employed for evaluating the performance of the 2nd order channels and are as follows:

- AmbiMeter (Polar Plot)
- AmbiMeter (Time Vs. Angle of Arrival)
- Directionally dependent frequency response analysis

5.1. The *AmbiMeter* Measures

The *AmbiMeter* measures were previously presented for 1st order signal evaluation in [10]. The term *AmbiMeter* decribes the name given to a specifically designed software application used for generating the resultant polar plots. For this work, in each case, the input signal must be a 2nd order B-Format RIR. The polar plot may only be used to assess the direct sound portion of the RIR while the Time Versus Angle (TVA) plot has been shown to be appropriate for inspecting the arrival of early reflections. The software operates by ambisonically decoding the signal to 144 equally spaced virtual loudspeaker directions on a circle using (7).

$$S_i = 0.5 \cdot ((2-d) \cdot W \cdot 1.4142) + d \cdot (\cos\theta_i \cdot X + \sin\theta_i \cdot Y + \cos 2\theta_i \cdot U + \sin 2\theta_i \cdot V) \quad (7)$$

Where S_i is the i^{th} speaker, θ_i is the angular position of that speaker and d is a directivity factor, varying between 0 and 1, as discussed in [12] [13]. The amplitude of the loudspeaker signals is then plotted and used to identify the direction from which the direct and early reflected sounds will arrive. For the polar plots the ampltude of the direct sound peak is plotted against the loudspeaker signal azimuth whereas in the TVA plots the amplitude at each time sample and loudspeaker angle is indicated using the gray scale plot. The results of the TVA plots are confirmed by using the ray tracing technique to accurately estimate the time and angle of arrival of wavefronts. Figure 8 illustrates the setup of the TVA tests and the associated early reflection predictions.

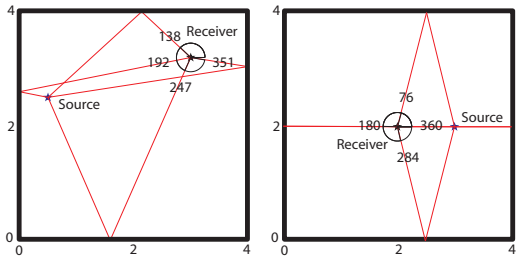


Figure 8: *Scaled images of the TVA test setup and resulting rays*

A 4m by 4m room and a 2D DWM are used to model the propogation of sound on the horizontal plane. Table 1 provides the estimated time and angle of arrival of the 1st order reflections labelled R1 to R4 in chronological order when the source and receiver is placed respectively at [0.5,2.5] and [3,3.2] for TVA test 1 and at [3,2] and [2,2] for TVA test 2. This is used as a measure for assessing that the receiver array encoding scheme in Figure 6 is correctly capturing the DWM modelled soundfield to the 2nd order. This is acheived by comparing the actual wavefront time and angle of arrival to the estimated one.

Test 1	D	R1	R2	R3	R4
Angle	196	137	191	350	246
Time(s)	0.0076	0.0099	0.0104	0.0133	0.0183
Test 2					
Angle	0/360	0/360	76	284	180
Time(s)	0.0029	0.0088	0.0121	0.0121	0.0147

Table 1: *Estimated times and angles of arrival of the first four reflections in Tests 1 and 2*

5.2. The Directionally Dependent Frequency Response

Directionally dependant frequency response analysis is used to assess how well the encoding technique matches the theoretical/ideal polar response as shown in Figure 7(a) for specific frequencies. The source signal is moved around the array of receivers at a constant distance and the U and V channels are calculated according to Figure 6(b). For a specific frequency the magnitude is plotted against the angle of the impulse sound source to provide a frequency dependent polar plot.

6. RESULTS ANALYSIS

The results for the *AmbiMeter* polar plot tests are presented in Figure 9. The 2nd order lobes point towards the direction of the sound

source which was placed at $0°, 90°, 180°$ and $270°$ in (a) and at $45°, 135°, 225°$ and $315°$ in (b) at 1 metre. This suggests that if the 2nd order B-Format signal was decoded to an appropriate loudspeaker array rather than a polar plot, the direct sound would be loudest at the desired angle.

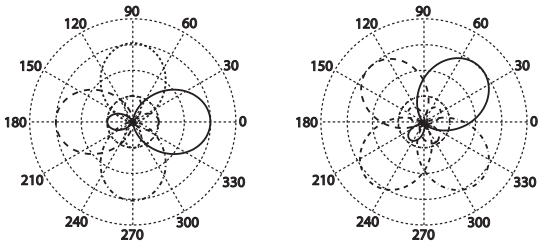


Figure 9: *AmbiMeter polar plot results for a sound source at 1m every* $45°$

In Figure 10 the results are presented for the first TVA test. The estimated time and angle of arrival for direct and reflected wavefronts are denoted by D and R1-R4. It may be observed that there are other reflections that are not predicted by the ray tracing technique in Table 1. Some of these occur before all of the 1st order reflections have arrived. Table 1 only estimates the 1st order reflections and these will not necessarily be the first four reflections to arrive at the microphone position. Therefore the other unlabelled wavefronts are 2nd order reflections and above, this will be demonstrated using Figure 11.

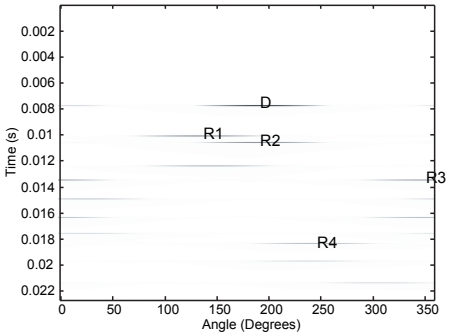


Figure 10: *Result for the first Time vs Angle AmbiMeter test*

Figure 11 is the second *AmbiMeter* TVA test result and in this case no 2nd order reflections arrive before R2 and R3. However R4 appears to arrive from all directions rather than $180°$. This is due to the fact that two other 2nd order reflections arrive at approximately the same time from different angles. This demonstrates how 2nd order reflections can seemingly obscure the estimated 1st order reflections. For clarity the time and angle of arrival values for the two suspected 2nd order reflections are calculated. Figure 12 illustrates the suspected movement of the 2nd order wavefronts. It is possible to easily evaluate the boundary location co-ordinates B1 - B2 for R5 and R6 in this simple room, and these are provided in Table 2. As the source and receiver positions are also known it is possible to use simple trigonometry to calculate the path length and angle of arrival for reflections 5 and 6. These are also presented in Table 2 and are placed onto the TVA test axis for completeness in Figure 13. In both cases it is clear that the predicted reflections arrive at the microphone position at the estimated time and angle of arrival.

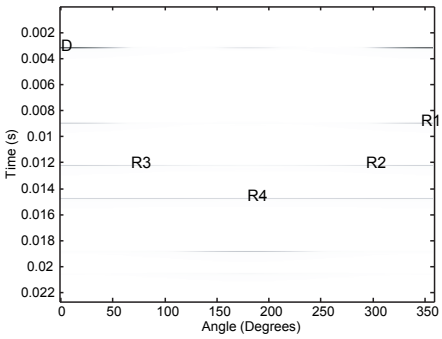


Figure 11: *Result for the second Time vs Angle AmbiMeter test*

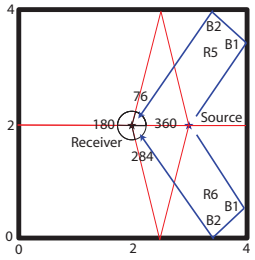


Figure 12: *Illustration of the path of two 2nd order reflections*

	Dist(m)	Arrival Time(s)	Angle	B1 [x,y]	B2 [x,y]
R5	5.2	0.0155	53	[4, 3.5]	[3.5, 4]
R6	5.2	0.0155	306	[4, 0.5]	[3.5, 0]

Table 2: *Path distance along with estimated time and angle of arrival for R5 and R6*

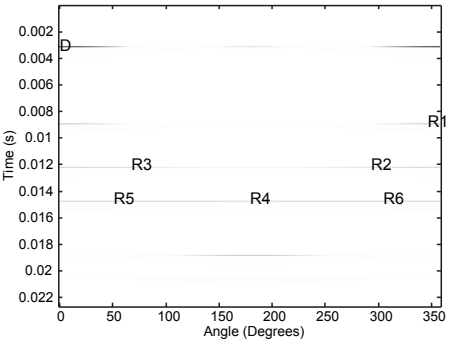


Figure 13: *Identification of the two 2nd order reflections*

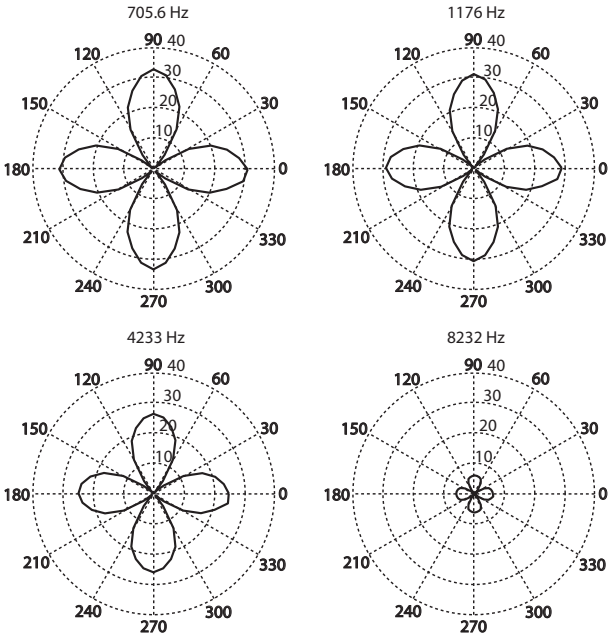


Figure 14: *U channel frequency dependent polar plots*

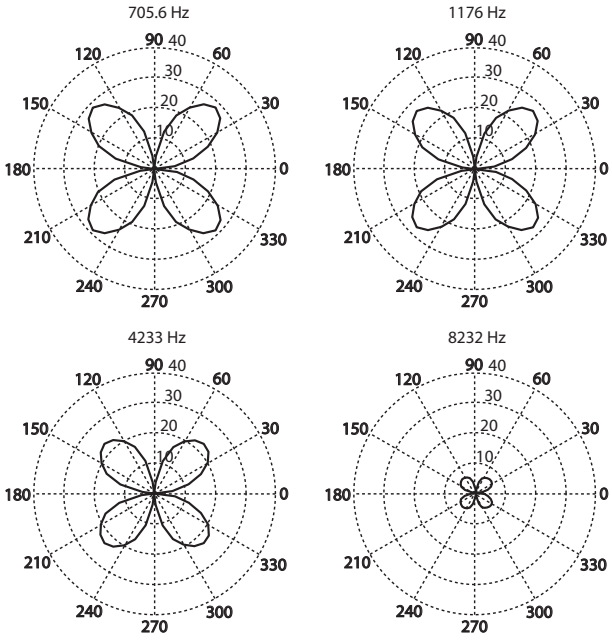


Figure 15: *V channel frequency dependent polar plots*

An indication of the actual sound pressure level is not necessary at this stage as the performance of the virtual microphone is not under test here. Rather the TVA tests help to provide a proof of concept in this work.

6.1. Directionally Dependent Frequency Response Results

The frequency dependent polar plots are presented for the both the U and V channels at four different frequencies chosen to sample the directional response across the usable range upto, but below, the spatial aliasing frequency. The spatial aliasing frequency is defined as the frequency whose wavelength is half the distance between the sampling points. Frequencies with smaller wavelengths are not supported and therefore must be filtered out in order to avoid ambiguity. The spatial aliasing frequency is defined in (8) as in [20] [22].

$$f_{sp} = \frac{c}{2 \cdot \Delta_{transducer}} \tag{8}$$

Where f_{sp} is the spatial alisaing frequency in Hertz, c is the speed of sound in air and $\Delta_{transducer}$ is the distance between the two pressure sensors/receivers in metres. For these tests a spacing of d = 0.04m was used (see Figure 6(a)) and so from (8) f_{sp} = 4.25kHz. The DWM model used a sampling frequency of 176.4kHz which corresponds to an internodal distance of 0.0027m. It should be noted that this sampling rate will support a smaller receiver spacing and hence a higher f_{sp} than that chosen. However issues arise as receiver positions can only exist at node positions which may or may not be at the numerically exact spatial co-ordinates associated with the virtual microphone array.

Figure 14 presents the frequency dependent polar plots for the U channel with Figure 15 showing the V channel response. Note that the radial axis is in dBs however the scale is not logarithmic for convenience. Figures 14 & 15 both indicate that the encoding techniques are approximately directionaly discriminating the incoming wavefronts at the correct azimuths in order to produce the desired polar response described by U and V in (3). It may be observed that theoretically the polar patterns for U and V should be identical but with a 45° rotation. Comparing the actual response for any pair of frequencies reveals that while the 45° rotation holds, the actual polar responses are not identical. The cause of this is the same reason that a smaller receiver spacing has not been chosen, the DWM is a discrete representation of a continuous medium. For example, when positioning two receivers with a distance of 0.02m apart in the DWM it is almost certain that these will not be exactly 0.02m apart. This is because the receiver position is 'snapped' to the closest DWM node position resulting in the apparent discrepancy. With relation to the proposed receiver arrangement, it is reasonable to assume that Crux A and B will introduce differing amounts of this discrepancy as they are orientated differently in the DWM. Therefore this issue may be improved if the DWM sampling frequency increases to infinity, which ensures that the distance between adjacent DWM nodes becomes infinitely small or alternatively the sound pressure between DWM nodes is accurately interpolated.

It is also apparent that the results are not consistent across the frequency spectrum, even when considering the few cases presented above. In addition, it might also be concluded that the polar response pattern worsens as frequency increases which is expected as the frequency wavelength reaches twice the receiver spacing [20] [23]. The relative gains of the virtual microphone also decreases as frequency increases. This apparent decrease may be attributed to two factors. The first being that as the frequency under test approaches the spatial aliasing frequency the anti-spatial aliasing filter will begin to reduce the gain. Secondly low frequencies are boosted for closer sound sources according to the proximity effect which is accentuated more as microphone order increases [23] [24]. Nevertheless the results are very encouraging and suggest that the proposed technique is working, although with some reser-

vations as to its performance with respect to frequency response. It should be pointed out that similar effects will occur with real microphone arrays as their sensitivities will not be identical [20].

7. CONCLUSIONS

This paper has proposed an encoding technique for the capture of 2nd order spherical harmonics in the horizontal plane for virtual acoustic models based on the digital waveguide mesh. The results have indicated that this initial encoding process is capable of effectivley sampling the virtual soundfield, although at this point the frequency response is still not ideal. The directionally dependent frequency response in conjunction with the *AmbiMeter* tests suggest that the calculated UV channels are able to provide an end listener with the directional characteristics of a DWM modelled acoustic space based on spherical harmonics to 2nd order. The frequency dependent polar plots also indicate that post-capture calibration must be carried out in order smooth out the main operational frequency band and this will be addressed in future work.

8. ACKNOWLEDGMENTS

This work has been supported by the University of York, Department of Electronics, EPSRC Doctoral Training Grant.

9. REFERENCES

[1] J.B. Allen & A. Berkley, "Image method for efficiently simulating small-room acoustics," *J.Acoustic Society of America*, vol. 75, no. 4, pp. 943–950, 1979.

[2] ODEON A/S, "Odeon room acoustics software," Available at http://www.odeon.dk, Accessed March 21, 2007.

[3] A. Farina, "Ramsete: Room acoustics modeling on pc," Available at http://pcfarina.eng.unipr.it/ramsete_ultimo/index.htm, Accessed March 20, 2007.

[4] A. Southern & D. Murphy, "Spatial encoding for digital waveguide mesh room modeling applications," in *Proc. of the AES 28th International Conference: The future of surround and beyond, Piteå, Sweden*, July 2, 2006, pp. 188–91.

[5] J.O. Smith, "A new approach to digital reverberation using closed waveguide networks," in *Proc. of 1985 International Computer Music Conference, Vancouver, Canada*, 1985, pp. 47–53.

[6] J.O. Smith, "Physical modeling using digital waveguides," *Computer Music Journal*, vol. 16, no. 4, pp. 74–91, 1992.

[7] S.A. Van Duyne & J.O. Smith, "Physical modeling with the 2-d digital waveguide mesh," in *Proc. of Int. Computer Music Conf, Tokyo, Japan*, 1993, pp. 40–47.

[8] D.T. Murphy, A. Kelloniemi, J. Mullen, and S. Shelley, "Acoustic modeling using the digital waveguide mesh," *IEEE Signal Processing Magazine*, vol. 24, no. 2, pp. 55–66, 2007.

[9] M.J. Beeson and D.T. Murphy, "Roomweaver: A digital waveguide mesh based room acoustics tool," in *Proc. of 7th International Conference of Digital Audio Effects (DAFx-04), Naples, Italy*, 2004, pp. 268–273.

[10] A. Southern, "Spatial rendering of digital waveguide mesh room acoustic models for multichannel sound," M.S. thesis, Department of Electronics, The University of York, UK, 2006, Available at http://www-users.york.ac.uk/~aps502/pubs.html.

[11] M.A. Gerzon, "With-height sound reproduction," *Journal of the Audio Engineering Society*, vol. 21, pp. 2–10, 1973.

[12] A. Farina, R. Glasgal, E. Armelloni, and A. Torger, "Ambiophonic principles for the recording and reproduction of surround sound for music," in *Proc. of 19th AES Conference on Surround Sound Techniques, Technology and Perception, Schloss Elmau, Germany*, 2001, pp. 21–24.

[13] B.J. Wiggins, *An Investigation into the real-time manipulation and control of three-dimensional sound fields*, Ph.D. thesis, University of Derby, UK, 2004, Available at http://sparg.derby.ac.uk/SPARG/PDFs/BWPhDThesis.pdf.

[14] R. Furse, "First and second order ambisonic decoding equations," Available at http://www.muse.demon.co.uk/ref/speakers.html, Accessed March 21, 2007.

[15] D. Malham, "Second and third order ambisonics - the furse-malham set," Available at http://www.york.ac.uk/inst/mustech/3d_Audio/secondor.html, Accessed March 21, 2007.

[16] M.A. Poletti, "A unified theory of horizontal holographic sound systems," *Journal of the Audio Engineering Society*, vol. 48, no. 12, pp. 1155–1182, 2000.

[17] F.J. Fahy, *Sound Intensity*, Elsevier Applied Science, London, 1989.

[18] J. Merimaa, "Applications of a 3d microphone array," in *Audio Engineering Society 112th Convention, Munich, Germany*, 2002.

[19] J. Merimaa, *Analysis, Synthesis and Perception of spatial sound - Binaural Auditory modeling and multichannel loudspeaker reproduction*, Ph.D. thesis, Helsinki University of Technology, 2006, Available at http://lib.tkk.fi/Diss/2006/isbn9512282917/.

[20] M.A. Gerzon, "Ultra-directional microphones - part 1," *Studio Sound*, pp. 434–437, October, 1970.

[21] P.S. Cotterell, *On the Theory of the Second-Order Soundfield Microphone*, Ph.D. thesis, University of Reading, 2002, Available at http://www.ambisonic.net/pdf/CotterellThesis/.

[22] J. Daniel, R. Nicol, and S. Moreau, "Further investigations of high order ambisonics and wavefield synthesis for holophonic sound imaging," in *Proc. of 114th AES Convention*, 2003.

[23] M.A. Gerzon, "Ultra-directional microphones - part 2," *Studio Sound*, pp. 501–504, November, 1970.

[24] M.A. Gerzon, "Ultra-directional microphones - part 3," *Studio Sound*, pp. 539–543, December, 1970.

HYPER-DIMENSIONAL DIGITAL WAVEGUIDE MESH FOR REVERBERATION MODELING

Antti Kelloniemi, *

Panphonics Oy,
Finland
antti.kelloniemi@panphonics.fi

Patty Huang,

CCRMA, Department of Music,
Stanford University, CA, USA
pph@ccrma.stanford.edu

Vesa Välimäki,

Lab. of Acoustics and Audio Signal Processing,
Helsinki University of Technology, Finland
vesa.valimaki@tkk.fi

Lauri Savioja,

Telecommunications Software and Multimedia Lab.,
Helsinki University of Technology, Finland
lauri.savioja@tkk.fi

ABSTRACT

Characteristics of digital waveguide meshes with more than three physical dimensions are studied. Especially, the properties of a 4-D mesh are analyzed and compared to waveguide structures of lower dimensionalities. The hypermesh produces a response with a dense and irregular modal pattern at high frequencies, which is beneficial in modeling the reverberation of rooms or musical instrument bodies. In addition, it offers a high degree of decorrelation between output points selected at different locations, which is advantageous for multi-channel reverberation. The frequency-dependent decay of the hypermesh response can be controlled using boundary filters introduced recently by one of the authors. Several hypermeshes can be effectively combined in a multirate system, in which each mesh produces reverberation on a finite frequency band. The paper presents two hypermesh application examples: the modeling of the impulse response of a lecture hall and the simulation of the response of a clavichord soundbox.

1. INTRODUCTION

The hyperdimensional digital waveguide (DWG) mesh is a 4-D version of the algorithm introduced by Van Duyne and Smith [1, 2]. There have been plenty of applications of the DWG mesh technique, but they have been limited to maximally three dimensions. 1-D digital waveguides are mostly used for simulating wave propagation in strings and tubes [3, 4, 5], 2-D meshes are applied in plate or membrane simulations [6, 7, 8, 9, 10], while rooms and resonant bodies of musical instruments are modeled with 3-D meshes [11, 12, 13, 9, 14, 15, 16]. Researchers have also modeled resonant objects and spaces with meshes having the number of dimensions different from that of the modeled object. For example, 2-D DWG meshes have been employed in room acoustic modeling [17, 18]. Recently, Mullen et al. have shown that a 2-D waveguide is an effective tool for simulating narrow acoustic tubes, which essentially contain a 1-D acoustic field [19].

* This work was funded by the Academy of Finland (project no. 201050) and the Nokia Foundation. The authors would like to extend their thanks to Dr. Tapio Lokki for the room impulse response measurement data, to Dr. Cumhur Erkut and Dr. Mikael Laurson for the clavichord recordings, and to Mr. Seppo Paulin for Fig. 4.

The idea of the hyperdimensional mesh was mentioned already in the original 2-D DWG mesh paper by Van Duyne and Smith [1], and later suggested again, for example, by Savioja et al. [11] and Rocchesso and Smith [20]. We recently published the first study about a 4-D DWG mesh [26]. In this paper, we investigate the properties of the 4-D mesh and possibilites opened by it through practical examples. Special emphasis is on employing the technique on reverberation modeling.

Artificial reverberation is widely used in musical performances and recordings. In addition to its use as an effect or in room acoustic simulation, reverberation modeling is needed in the synthesis of musical instruments with a resonating body, such as the soundbox or soundboard of stringed keyboard instruments.

This paper is organized as follows. Section 2 discusses the normal modes of vibration in enclosed spaces. In Section 3, the hyperdimensional DWG mesh, or the hypermesh for short, is discussed as an extension to the previously known waveguide mesh methods. Section 4 describes the application of the hypermesh in two cases of artificial reverberation: simulation of a lecture hall's impulse response and simulation of the soundbox of a musical keyboard instrument.

2. NORMAL MODES IN ENCLOSURES WITH RIGID BOUNDARIES

Sound pressure waves reflect from boundaries, such as walls and furniture in a room. When the time interval between successively received reflected sounds is short, they are perceived as reverberation instead of individual echoes. In any closed space, sound is reflected along multiple closed propagation paths, and thus standing waves occur. The standing waves determine the modal structure in the frequency response of the acoustic system. At low frequencies, the modes are sparsely spaced in frequency, but at frequencies above a critical frequency, often called the Schroeder frequency, the modal peaks are not distinguished individually by the ear [21, 22]. If the modal density created by an artificial reverberation algorithm is too low in this high frequency region, tonality or a metallic timbre is perceived.

Sound pressure between two rigid boundaries located at $x = 0$

and $x = L$ must fulfill the boundary condition

$$\frac{dp}{dx} = 0. \tag{1}$$

The sound pressure value at a certain modal frequency at any point is given by a solution for (1) that can be written as

$$p(x) = A\cos(k_n x), \tag{2}$$

where A is an arbitrary coefficient, $k_n = n\pi/L$, and $n = 0, 1, 2, \ldots$ is the integer index of the current mode along dimension x having corresponding length L [22].

The 1-D solution can be extended to N dimensions, where the sound pressure value at point $(x_1, x_2, \ldots, x_N)$ inside the N-dimensional rectangular space at a certain modal frequency is

$$p_{n_1 n_2 \ldots n_N}(x_1, x_2, \ldots, x_N) = B \prod_{i=1}^{N} \cos(k_{n_i} x_i), \tag{3}$$

where B is an arbitrary coefficient and n_i are the mode index numbers for each dimension. The modes appear at frequencies

$$f_{n_1 n_2 \ldots n_N} = \frac{c}{2\pi} k_{n_1 n_2 \ldots n_N}, \tag{4}$$

where c is the sound velocity. The constant $k_{n_1 n_2 \ldots n_N}$ is a combination of all k_{n_i}:

$$k_{n_1 n_2 \ldots n_N} = \pi \left[\sum_{i=1}^{N} \left(\frac{n_i}{L_i} \right)^2 \right]^{1/2}, \tag{5}$$

where L_i is the spatial length along ith dimension.

The first axial standing wave along each dimension with $n_i = 1$ has a frequency whose corresponding wavelength is equal to twice the trajectory length. Other standing waves on the same trajectory are created at multiples of this base frequency. In addition to these 1-D modes occurring between parallel boundaries, multi-dimensional standing waves are supported as closed propagation paths are created between multiple boundaries. For these diagonal and oblique modes, two or more indices have values above zero, respectively.

The modal frequencies are inversely proportional to the trajectory length, so, for example in large halls the modes start from lower frequencies than in small rooms. The modal density also increases with frequency, as suggested by (4) and (5). Actual rooms and halls are not perfectly rectangular and have furniture and other objects affecting sound propagation. So the trajectory lengths are not equal at all frequencies. More propagation trajectories are supported, especially at high frequencies, resulting in an even denser and inharmonic modal structure. In more complex shapes, such as fan-shaped rooms or bodies of musical instruments, the modal structure is too complex to be managed in closed form expressions. Instead, numerical approximations of the wave propagation are needed.

3. DIGITAL WAVEGUIDE MESH METHOD

The DWG mesh provides a computational model for multi-dimensional wave propagation. It was created as an extension of 1-D digital waveguides popular in the physical modeling-based sound synthesis applications [1, 23]. A 1-D DWG consists of two delay lines passing signals into opposite directions and scattering junctions between the delay lines. The input signal to a junction can be passed through, partially transmitted, or reflected back.

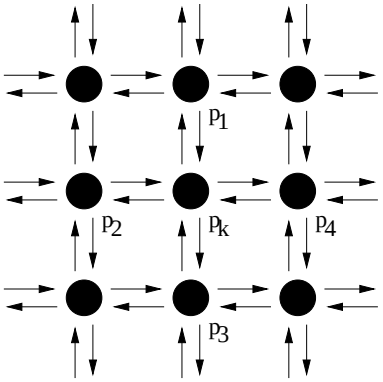


Figure 1: *A two-dimensional rectilinear DWG mesh structure. p_k is the junction currently calculated and p_l, where $l = 1, 2, 3,$ or 4, are its axial neighbors as in (8).*

The mesh can be constructed in various ways. The choice can be made between two different variable types and multiple topologies. Common to all DWG mesh schemes is the regular discretization, both in time and in space.

3.1. The digital waveguide mesh updating functions

A DWG mesh consists of bidirectional delay lines and scattering junctions connecting them at regular nodal points. For example, a 2-D DWG mesh structure is shown in Fig. 1. In a homogeneous N-dimensional mesh each junction has $2N$ neighbors, and all interconnections have equal impedances. If the delays are located at the interconnections of the nodes, the updating function of each junction is written as

$$p_k(n) = \frac{2}{N} \sum_l p_l^+(n), \tag{6}$$

where $p_l^+(n)$ are the incoming wave variable values of each interconnection l of the current junction at time instant n. The outgoing values are then updated using the current junction value p_k:

$$p_l^-(n) = p_k(n) - p_l^+(n). \tag{7}$$

The outgoing values are transformed into ingoing values of neighboring junctions when they are passed by the unit delays in the interconnections during the next computational time step. This formulation of the mesh is called the wave variable formulation, or W mesh.

Another formulation of the same functionality uses physically measurable variables instead of their traveling wave decomposition, as used in (6) and (7). In the so-called Kirchhoff formulation, or K mesh, the delays are located at the nodal points of the mesh, and the updating function of each junction is written as

$$p_k(n) = \frac{\sum_l p_l(n-1)}{N} - p_k(n-2), \tag{8}$$

where p_l are now the values of the neighboring junctions. While being numerically less robust, in multi-dimensional models this formulation requires considerably less main system memory than an equivalent W mesh [24].

The sampling frequency is related to the dimensionality N of the mesh by

$$f_s = \frac{c\sqrt{N}}{\Delta x}, \tag{9}$$

where c is the wave propagation speed in the mesh and Δx is the spatial sampling interval corresponding to the distance between two neighboring junctions [13]. The practical frequency bandwidth for the mesh depends on its geometric topology [25]. For example, a triangular mesh cannot produce resonances above $f_s/3$ and a rectilinear mesh has a spectrum that mirrors itself at $f_s/4$. Due to the mirroring of resonances around half the Nyquist limit, the output of a rectilinear mesh is usually lowpass filtered in order to retain only the "unique" modes below $f_s/4$. However, this filtering is not required if the user is only interested in having an output with a maximal number of modes instead of an exact physical model of a resonating structure.

In a DWG mesh, the number of degrees of freedom of the model is equal to the number of delay elements. So in a homogeneous and freely resonating rectilinear K mesh the maximum number of modes below the mirroring frequency is equal to the number of junctions. The highest mode index number n_i is equal to the number of junctions along the corresponding dimension. If the phase of the reflected wave is preserved at the boundary, $n_i \geq 0$. In the case of phase reversing reflection, the lowest modes are canceled, and only modes with $n_i \geq 1$ are supported.

3.2. Hyperdimensional DWG mesh structure

The mesh dimensionality N is not restricted to the limits of our physical world. Instead, hyperdimensional meshes are easy to construct by adding more interconnections between the scattering junctions [20, 13, 26].

As seen in Fig. 2, the modes are distributed equally in a 1-D DWG structure. In rectilinear meshes with higher dimensionalities, the modes are densest near $f_s/4$ and sparsest at frequencies close to DC and $f_s/2$. As the number of junctions is kept constant with increasing dimensionality, the number of junctions along each dimension is diminished. This packs the modes closer around $f_s/4$. At the same time, the modal frequencies become higher because the sampling frequency increases with dimensionality, as seen from (9). The number of independent indices n_i in (5) is equal to the number of dimensions. Maximizing the number of dimensions and choosing the L_i values close together from a prime number series minimizes the harmonicity of the mode distribution, which is beneficial for simulating reverberation over a wide frequency bandwidth [27]. The inharmonicity is further augmented by perturbation in mode frequencies caused by the numerical dispersion inherent in the waveguide mesh structure [2, 8].

3.3. Boundary conditions

For realistic reverberation modeling, frequency-dependent losses have to be implemented. In real rooms, high frequencies usually decay faster than low frequencies due to absorption of energy by air and wall materials. Another important feature is the strong modal frequencies with long decay times characterizing some spaces, especially musical instrument bodies.

In a DWG mesh, the frequency-dependent losses can be combined and implemented with boundary filters. In this way the inner mesh structure is kept lossless and homogeneous. As discussed earlier, the inner mesh was implemented using K formulation. As

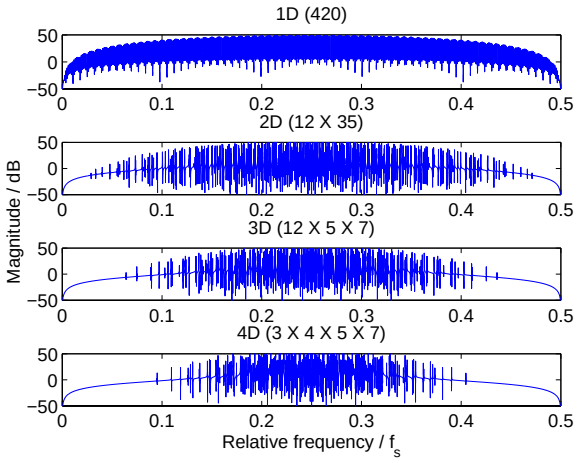


Figure 2: *Frequency responses of rectilinear meshes with 420 scattering junctions, organized in four different dimensionalities. The reflection coefficient of all boundaries was $R = -1$. The mesh was initialized at a corner junction, and the output was read at the same location.*

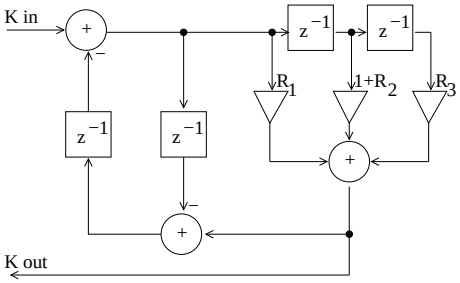


Figure 3: *The boundary junction with KW-conversion and second-order FIR filter [28].*

boundary filters are easier to design for traveling wave variables, a boundary junction structure including a variable-type converter and a second-order FIR filter was used, as depicted in Fig. 3 [28].

The boundary reflection characteristics are determined with coefficients R_1, R_2, and R_3. This kind of low-order filter can model simple lowpass behavior and is effective enough for nonexact simulation of impulse responses.

4. APPLICATION TO ARTIFICIAL REVERBERATION

The frequency response of a reverberant structure can be coarsely divided into two bands. At low frequencies, the modes can be individually heard. Their frequencies and decay times are psychoacoustically important, so recreating them exactly is needed for convincingly simulating the response. At higher frequencies, the modal frequencies are not heard individually, and thus an exact physical model is not needed. For natural sounding simulation of high-frequency reverberation, the key issues are the density and the irregularity of the modal structure [27].

In the presented examples, the hypermesh structure is used to generate the high-frequency portion of the impulse responses

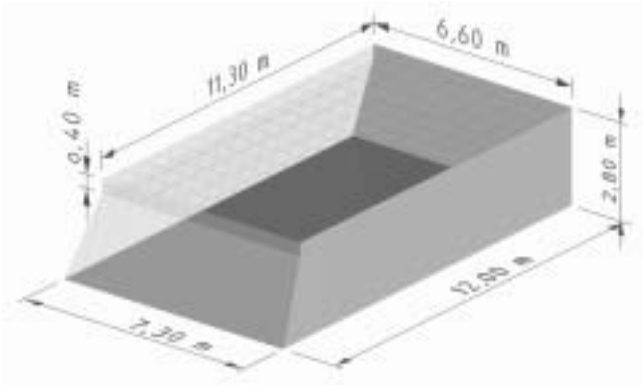


Figure 4: *Dimensions of the lecture hall T3 at the Helsinki University of Technology.*

of a lecture hall and a clavichord soundbox. In both cases, the high-frequency hypermesh response is combined with the output of other resonator models providing the low-frequency modes at physically correct frequencies. This is similar to hybrid models which have been used for simulating bodies of musical instruments, in which a reverberation algorithm is combined with a resonator bank in parallel [14, 29].

Sound samples of the studied cases are available at http://www.acoustics.hut.fi/∼vpv/publications/hypermesh/.

4.1. Case: Lecture hall

In the first example, the hypermesh is applied to the simulation of room reverberation. The impulse response of lecture hall T3 at the Helsinki University of Technology was measured and used as a reference. The dimensions of the lecture hall are shown in Fig. 4. The ceiling area is smaller than the floor area, as the left-side wall and the back wall are sloping. A soft absorbent plate is hung at 0.40 m below the ceiling, covering the full area from the back wall to 3.0 m from the front wall.

The room impulse response was measured 5.5 m from the left wall and 10.0 m from the front wall at a height of 1.7 m. The speaker used as a sound source was located 2.7 m away from both the left and the front wall equally, at a height of 1.2 m. The five most prominent frequency modes and their corresponding 60 dB decay times are listed in Table 1. They were evaluated from the measured signal and then removed from it.

A two-pole, two-zero inverse filter was designed for mode removal after determining the frequencies and decay times. The T_{60}s were estimated by fitting a straight line to the time-domain values of the response's energy decay relief (EDR) [30] at the frequency bin associated with a particular mode. The slope of the line was then inverted and scaled to obtain the T_{60} value. The zeros of each inverse filter are complex conjugates whose angles are the positive and negative radian frequency of the mode to be removed and whose radius R is matched to the T_{60} of the same mode by the relationship $R = e^{\frac{\ln(0.001)}{T_{60} f_s}}$. The two poles are identical to the zeros except that the radius is slightly contracted by a factor very close to 1, in order to isolate the effect of the zeros to the target mode [31]. 0.9999 was used when neighboring modes are very close to each other, but a factor of 0.999 was sufficient in most cases. The reverberation time of the remaining signal was evaluated on octave

Frequency (Hz)	T_{60} (sec)	Magnitude (dB)
75.96	0.8134	-25.90
102.52	1.3244	-30.58
122.02	1.4195	-29.31
138.08	0.9622	-31.85
148.02	1.3221	-29.02

Table 1: *Analysis results for prominent modes of the lecture hall impulse response, to be implemented with a resonator bank.*

Boundary	R_1, R_3	R_2
3D hard walls	0.01958	0.84000
3D soft ceiling	0.00896	0.85918
4D mid frequencies	0.00440	0.67470
4D high frequencies	0.00200	0.91200

Table 2: *Filter coefficients of the room impulse response simulation.*

bands, and these T_{60} values were used as the optimization goal for the 4-D meshes.

The five highest octave bands with central frequencies from 1 kHz to 16 kHz were simulated by a multirate system consisting of two rectilinear 4-D meshes of $7 \times 8 \times 10 \times 13$ junctions each. The hypermesh dimensions were chosen to produce maximally dense and irregular modal pattern over a sufficiently wide frequency band. A multirate system was utilized, as use of large meshes and filter structures was thus avoided. This is essential, as computational cost of a hyperdimensional mesh grows rapidly with the number of junctions. The boundary filters shown in Fig. 3 were implemented at one end of the longest dimension, while perfectly reflecting, phase inversing conditions were implemented at other boundaries by fixing their values to zero. The simulation was run for 48000 time steps for the first mesh, and for 12000 steps for the second mesh, as its output was upsampled by a factor of 4. A Nyquist filter was used for anti-aliasing. The coefficients of the 3-tap FIR filters at one $7 \times 8 \times 10$ junction boundary of each mesh were optimized for minimizing the maximum error in T_{60}-values of the combined output. A Nelder-Mead optimization, provided by Matlab, was used.

At lower frequencies, the exact frequencies of each mode are perceptually important, and thus simulating only the decay times would not provide an appropriate result. Instead, a 3-D triangular mesh, also known as a 3-D dodecahedral or hexagonal close packed, was defined to model the low-frequency response. The low-frequency mesh topology was chosen by the fact that the dense triangular mesh exhibits minimal numerical errors. The mesh dimensions were designed to match the room dimensions as closely as possible with junction spacing of 0.2 m. Using (9), the sampling frequency of the mesh is seen to be $f_s \approx 2.9$ kHz and the highest frequency modeled is thus about 950 Hz. Linear-phase FIR filters were designed for all boundaries of the mesh. Two different sets of filter coefficients were optimized to match the reverberation times of the received signal to the measurement results. One filter was used for the soft ceiling, another for other surfaces. The coefficients used are listed in Table 2. The two hypermeshes were excited at a corner junction with impulse responses of high-order filters to match their frequency responses together at crossover

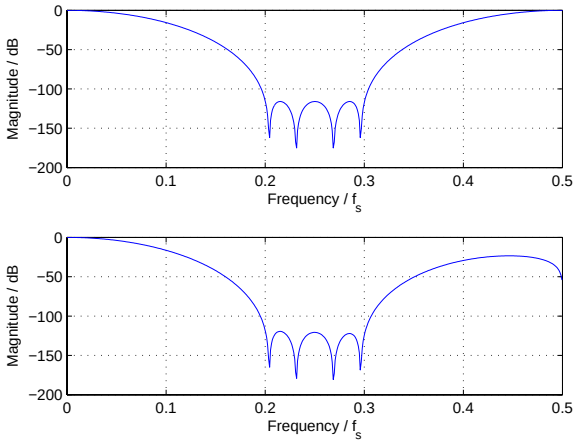


Figure 5: *Frequency responses of the input filters of the two hypermeshes used in the room impulse response simulation: above for the mid frequencies and below for the highest frequencies.*

Frequency (Hz)	Measured T_{60}	Simulated T_{60}
125	1.4640	0.5116
250	0.7403	0.6437
500	0.6503	0.4493
1000	0.5833	0.4277
2000	0.5640	0.5146
4000	0.5221	0.5212
8000	0.4739	0.4876
16000	0.3526	0.3388

Table 3: *Reverberation times of the measured and simulated room impulse responses of the lecture hall T3. Center-frequencies of the octave bands are listed.*

frequencies. The frequency responses of the filters are shown in Fig. 5. The 3-D mesh was excited with a unit impulse at a location closely matching the location of the sound source in the reference measurement setup and the response was recorded from a point representing the measurement location, respectively.

The measured and simulated T_{60} values are shown in Table 3, and the time-frequency representations of the responses can be seen in Figs. 6 and 7, respectively. The resulting responses can be seen to be a good match in terms of reverberation times, especially at the frequencies above 1 kHz modeled specifically by the hypermeshes. The initial shapes of the frequency responses are significantly different, as the equalization of the relative magnitudes of the mesh outputs was set only by ear. This affects the first 0.1 seconds of the response. The difference in the latter part of the signals is explained by the noise present in the measured response in Fig. 6 and absent in the simulated response in Fig. 7. Also, for a better match at frequencies below 1 kHz, the low-frequency model could have been implemented more precisely especially in terms of spatial resolution and boundary filter design. However, this was not the focus of this paper.

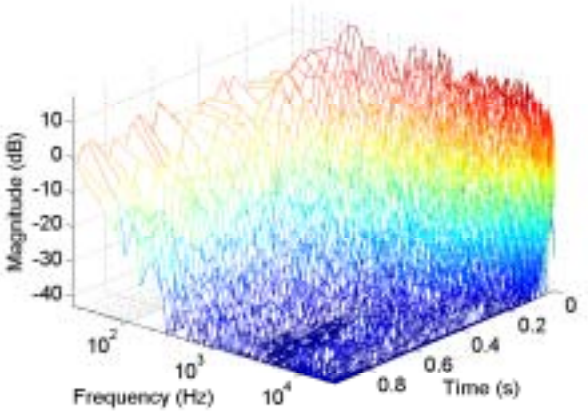


Figure 6: *Measured impulse response of the lecture hall.*

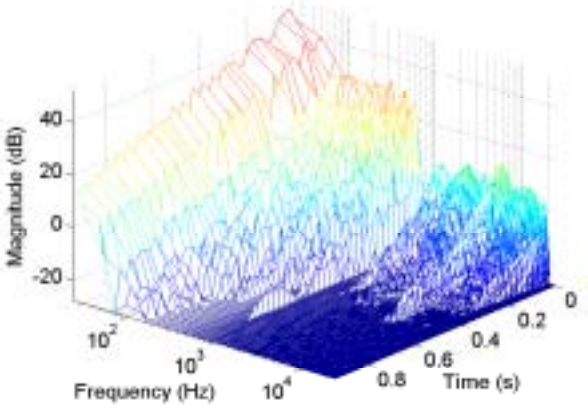


Figure 7: *Simulated impulse response combined from the outputs of a 3-D triangular mesh for low frequencies and two 4-D hypermeshes for the frequencies above 700 Hz.*

4.2. Case: Clavichord soundbox

In the second example, the reverberant impulse response of a clavichord soundbox was modeled using two hypermeshes and a resonator bank. The response used as the overall simulation target was produced with an impulse hammer impact on the soundbox while the strings of the clavichord were carefully damped. Figure 8 shows the time-frequency representation of the impulse response of the clavichord soundbox. It contains many modes between about 30 Hz and 3 kHz, but no significant energy at frequencies higher than that.

The hypermeshes served to generate an approximation of the dense high-frequency modes of the soundbox. The hypermesh simulation target was the soundbox impulse response whose long-ringing modes in the low-frequency range were removed by inverse filtering [31]. 28 biquadratic resonators were used to isolate the prominent modes below 500 Hz. Fewer resonators can be used in practice depending on the desired synthesis quality.

A multirate system was created, as in Section 4.1, to implement the soundbox reverberator in Fig. 9. Only two hypermeshes of $8 \times 9 \times 11 \times 13$ junctions were needed to generate sufficiently dense reverberation within the reduced bandwidth. The output of one of the meshes was upsampled by a factor of 3. The total response was filtered with a sixth-order LPC filter, whose coefficients were obtained from the measured soundbox impulse response after extracting the most prominent modes. Boundary filters were designed to match the decay times which were analyzed for each one third-octave band. The meshes were initialized with a filter output signal having a frequency response as depicted in Fig. 10 for flattening the overall shape of the mesh response seen in Fig. 2. Responses were recorded at opposite corner locations compared to the initialization points.

Figure 11 is the time-frequency representation of the hypermesh model of the high-frequency response of the clavichord soundbox. The low-frequency modes that have been extracted are not included in this model. It is seen by comparing Figs. 8 and 11 that the hypermesh model produces a similar, but not exactly identical, response between about 100 Hz and 2 kHz. The low-frequency modes need to be implemented with separate resonators to obtain a full model of the soundbox.

The commuted synthesis [32, 33] clavichord model described in an article by Välimäki et al. [34] can be enhanced by replacing the sampled soundbox response triggered at each note with a soundbox reverberation module, such as the one described above. In a synthesis model using this scheme, the output of a string module would be fed into a body/resonator module as shown in Fig. 9. A similar solution has been used for sound synthesis of the harpsichord, where reverberation from the soundboard was simulated with a feedback delay network reverberator [35].

A reverberator model with a spatial interpretation such as the hypermesh supports multiple input locations and allows for subtle differences in reverberation for each note. This would be analogous to subtle differences in the soundbox response resulting from each string's unique driving point on the bridge. The contribution of a hypermesh reverberator to a synthetic clavichord tone makes it sound more realistic and lively than if it were only overlaid with a sampled soundbox impulse response, which always adds the same reverberation effect to the tone.

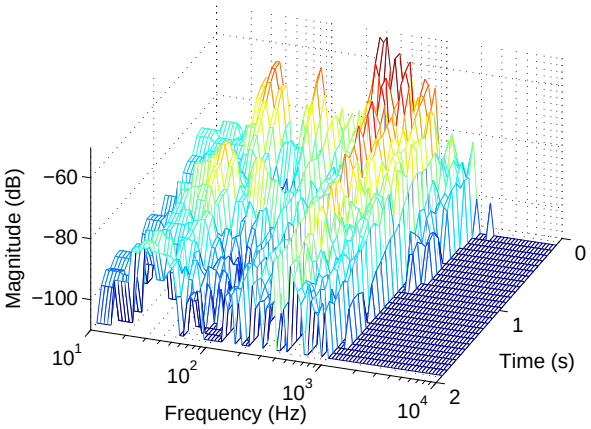


Figure 8: *Measured impulse response of the clavichord soundbox.*

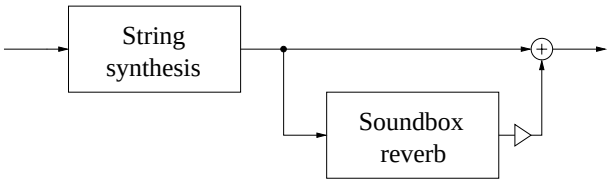


Figure 9: *Simplified clavichord synthesis model incorporating a reverberation module to simulate the soundbox.*

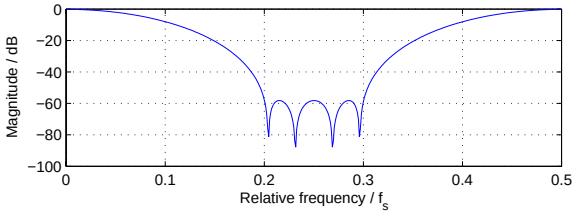


Figure 10: *Frequency response of the input filter of the hypermeshes used in the clavichord soundbox impulse response simulation.*

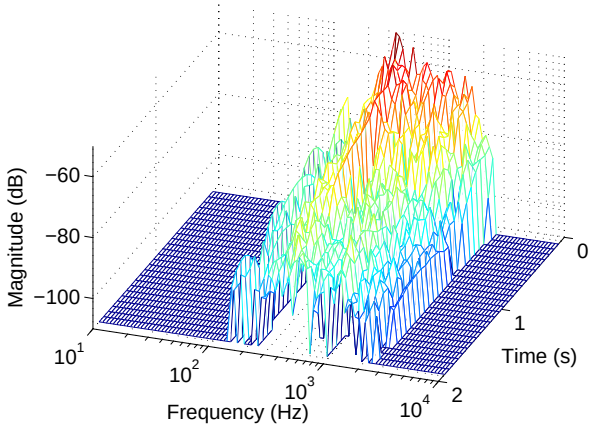


Figure 11: *Combined response of the two hypermeshes used in the simulation of the clavichord soundbox impulse response. The low-frequency modes implemented with separate resonators are not included.*

5. CONCLUSIONS AND FUTURE WORK

The hyperdimensional digital waveguide mesh discussed in this paper is a four-dimensional variation of the DWG mesh technique. The main advantage of this structure is its ability to provide a much more dense and irregular modal structure at high frequencies compared to meshes of lower dimensionality. This encourages the utilization of the presented technique, for example in the creation of artificial reverberation as presented in our paper. The attenuation characteristics of the mesh can be controlled by similar boundary conditions as used with 2-D and 3-D meshes, thus enabling shaping of the resulting magnitude response.

This paper has shown two applications: simulation of a lecture hall and simulation of a clavichord soundbox. The quality obtained in these simple examples encourages to study further the uses of hyperdimensional meshes in the field of spatial audio. In the future, the hypermesh should be compared with other methods and listening tests should be performed to assess the sound quality.

6. REFERENCES

[1] S. A. Van Duyne and J. O. Smith III, "Physical modeling with the 2-D digital waveguide mesh," in *Proc. Int. Computer Music Conf.*, Tokyo, Japan, Sept. 1993, pp. 40–47.

[2] S. A. Van Duyne and J. O. Smith III, "The 2-D digital waveguide mesh," in *Proc. IEEE WASPAA*, New Paltz, NY, USA, Oct. 1993, pp. 17–20.

[3] J. O. Smith and G. P. Scavone, "The one-filter Keefe clarinet tonehole," in *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, NY, October 1997.

[4] M. Karjalainen, V. Välimäki, and T. Tolonen, "Plucked-string models: from the Karplus-Strong algorithm to digital waveguides and beyond," *Computer Music Journal*, vol. 22, no. 3, pp. 17–32, Fall 1998.

[5] J. Bensa, S. Bilbao, R. Kronland-Martinet, and J. O. Smith, "The simulation of piano string vibration: from physical models to finite difference schemes and digital waveguides," *J. Acoust. Soc. Am.*, vol. 114, no. 2, pp. 1095–1107, 2003.

[6] F. Fontana and D. Rocchesso, "Physical modeling of membranes for percussion instruments," *Acustica united with acta acustica*, vol. 84, pp. 529–542, 1998.

[7] J. Laird, P. Masri, and C. N. Canagarajah, "Efficient and accurate synthesis of circular membranes using digital waveguides," in *Proc. IEE Colloquium on Audio and Music Technology: The Challenge of Creative DSP*, November 1998, pp. 12/1–12/6.

[8] L. Savioja and V. Välimäki, "Reducing the dispersion error in the digital waveguide mesh using interpolation and frequency-warping techniques," *IEEE Trans. Speech and Audio Processing*, vol. 8, no. 2, pp. 184–194, Mar. 2000.

[9] M. L. Aird, L. Laird, and J. ffitch, "Modelling a drum by interfacing 2-D and 3-D waveguide meshes," in *Proc. Int. Computer Music Conf.*, Berlin, Germany, Aug. 2000, pp. 82–85.

[10] S. Bilbao, "Sound synthesis for nonlinear plates," in *Proc. Int. Conf. Digital Audio Effects (DAFx)*, Madrid, Spain, September 2005, pp. 243–248, http://dafx05.ssr.upm.es/Proc_DAFx05/P_243.pdf.

[11] L. Savioja, T. Rinne, and T. Takala, "Simulation of room acoustics with a 3-D finite difference mesh," in *Proc. ICMC'94*, Aarhus, Denmark, Sept. 1994, pp. 463–466.

[12] L. Savioja, J. Backman, A. Järvinen, and T. Takala, "Waveguide mesh method for low-frequency simulation of room acoustics," in *Proc. 15th Int. Congr. Acoust. (ICA)*, Trondheim, Norway, June 1995, vol. 2, pp. 637–640.

[13] L. Savioja, J. Huopaniemi, T. Lokki, and R. Väänänen, "Creating interactive virtual acoustic environments," *J. Audio Eng. Soc.*, vol. 47, no. 9, pp. 675–705, Sept. 1999.

[14] P. Huang, S. Serafin, and J. O. Smith, "A 3-D waveguide mesh model of high-frequency violin body resonances," in *Proc. Int. Computer Music Conf.*, Berlin, Germany, Aug. 2000, pp. 86–89.

[15] M. J. Beeson and D. T. Murphy, "Virtual room modelling using hybrid digital waveguide mesh techniques," in *Proc. 147th meeting of ASA*, New York, USA, May 2004.

[16] G. R. Campos and D. M. Howard, "On the computational efficiency of different waveguide mesh topologies for room acoustic simulation," *IEEE Trans. Speech and Audio Processing*, vol. 13, no. 5, pp. 1063–1072, Sept. 2005.

[17] D. T. Murphy and D. M. Howard, "Modelling and directionally encoding the acoustics of a room," *Electronics Letters*, vol. 34, no. 9, pp. 864–865, Apr. 1998.

[18] A. Kelloniemi, V. Välimäki, and L. Savioja, "Simulation of room acoustics using 2-D digital waveguide meshes," *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, Toulouse, France, May 2006, vol. 5, pp. 313–316.

[19] J. Mullen, D. M. Howard, and D. T. Murphy, "Waveguide physical modeling of vocal tract acoustics: Flexible formant bandwidth control from increased model dimensionality," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 14, no. 3, pp. 964–971, May 2006.

[20] D. Rocchesso and J. O. Smith, "Circulant andelliptic feedback delay networks for artificial reverberation," *IEEE Trans. Speech and Audio Processing*, 5(1), Jan. 1997, pp. 51–63.

[21] A. D. Pierce, *Acoustics, 2nd ed.* McGraw-Hill, New York, 1989, pp. 293–294.

[22] H. Kuttruff, *Room Acoustics, 4th ed.* Spon Press, London, 2000.

[23] V. Välimäki, J. Pakarinen, C. Erkut, and M. Karjalainen, "Discrete-time modelling of musical instruments," *Reports on Progress in Physics*, vol. 69, no. 1, pp. 1–78, January 2006, http://www.iop.org/EJ/abstract/0034-4885/69/1/R01/.

[24] M. Karjalainen and C. Erkut, "Digital waveguides vs. finite difference schemes: Equivalence and mixed modeling," *EURASIP J. Applied Signal Process.*, no. 7, pp. 978–989, June 2004.

[25] F. Fontana, D. Rocchesso, "Signal-theoretic characterization of waveguide mesh geometries for models of two-dimensional wave propagation in elastic media," *IEEE Trans. Speech and Audio Process.*, vol. 9, no. 2, pp. 152–161, Feb. 2001.

[26] A. Kelloniemi, V. Välimäki, P. Huang, and L. Savioja, "Artificial reverberation using a hyper-dimensional FDTD mesh," in *Proc. European Signal Processing Conference (EUSIPCO)*, Antalya, Turkey, September 2005, http://signal.ee.bilkent.edu.tr/defevent/papers/cr1422.pdf.

[27] M. Karjalainen and H. Järveläinen, "More about this reverberation science: Perceptually good late reverberation," *AES 111th Convention*, New York, NY, USA, Nov. 2001, preprint no. 5415.

[28] A. Kelloniemi, "Frequency-dependent boundary condition for the 3-D digital waveguide mesh," *Int. Conf. Digital Audio Effects (DAFx)*, Montreal, Canada, Sept. 2006.

[29] H. Penttinen, M. Karjalainen, T. Paatero, and H. Järveläinen, "New techniques to model reverberant instrumentbody responses," in *Proc. ICMC'01*, Havana, Cuba, Sept. 2001, pp. 182–185.

[30] J.-M. Jot, "An analysis/synthesis approach to real-time artificial reverberation," in *Proc. ICASSP 1992*, San Francisco, California, U.S.A., Mar. 1992, vol. 2, pp. 221–224.

[31] J. O. Smith III, "Physical audio signal processing: for virtual musical instruments and digital audio effects," online book at http://ccrma.stanford.edu/~jos/pasp/, Center for Computer Research in Music and Acoustics (CCRMA), Stanford University.

[32] J. O. Smith III, "Efficient synthesis of stringed musical instruments," in *Proc. ICMC'93*, Tokyo, Japan, Sept. 1993, pp. 64–71.

[33] M. Karjalainen and V. Välimäki, "Model-based analysis/synthesis of the acoustic guitar," in *Proc. Stockholm Music Acoustics Conf.*, Stockholm, Sweden, Jul.-Aug., 1993, pp. 443–447.

[34] V. Välimäki, M. Laurson, and C. Erkut, "Commuted Waveguide Synthesis of the Clavichord," *Computer Music J.*, 27(1), pp. 71–82, Spr. 2003.

[35] V. Välimäki, H. Penttinen, J. Knif, M. Laurson, and C. Erkut, "Sound synthesis of the harpsichord using a computationally efficient physical model," EURASIP J. on Applied Signal Processing, no. 7, pp. 934–948, June 2004. Special Issue on Model-Based Sound Synthesis.

RAY ACOUSTICS USING COMPUTER GRAPHICS TECHNOLOGY

Niklas Röber, Ulrich Kaminski and Maic Masuch

Games Research Group
Department of Simulation and Graphics,
Otto-von-Guericke-University Magdeburg, Germany

niklas@isg.cs.uni-magdeburg.de

ABSTRACT

The modeling of room acoustics and simulation of sound wave propagation remain a difficult and computationally expensive task. Two main techniques have evolved, with one focusing on a real physical - wave-oriented - sound propagation, while the other approximates sound waves as rays using raytracing techniques. Due to many advances in computer science, and especially computer graphics over the last decade, interactive 3D sound simulations for complex and dynamic environments are within reach.

In this paper we analyze sound propagation in terms of acoustic energy and explore the possibilities to map these concepts to radiometry and graphics rendering equations. Although we concentrate on ray-based techniques, we also partially consider wave-based sound propagation effects. The implemented system exploits modern graphics hardware and rendering techniques and is able to efficiently simulate 3D room acoustics, as well as to measure simplified personal HRTFs through acoustic raytracing.

1. INTRODUCTION

Physically correct sound simulations of larger and more complex environments remain a difficult, if not impossible task. This is mainly due to the extensive nature of sound wave propagation, along its complex interaction with scene objects. Unlike light, the audible spectrum covers a large area of frequency bands (octaves), and is additionally, due to a slow propagation, highly time-dependent. Although, this introduces several complications, it also allows, in certain situations, to discard some of the wave phenomena, especially for the higher frequency bands. As a result, two main approaches have evolved for the simulation of sound wave propagation: The wave-based and the ray-oriented techniques, with the first one concentrating on the lower and the last one on the middle and higher frequency ranges. Here Section 2 has a closer look on both techniques and compares them in terms of efficiency and applicability. Although several improvements have been reported for both techniques, sound simulations are in general performed offline and are valid only for certain frequency ranges. Due to advances in computational power, as well as in computer graphics and acoustics, interactive and dynamic ray-based sound simulations are feasible also for complex and more difficult scenes.

Accelerated and driven by computer games and the demand for an even higher visual realism, computer graphics hardware has evolved tremendously over the last decade and nowadays outperforms the CPU in terms of computational capacity by several magnitudes. As of the easy availability of this processing power, graphics hardware has been exploited in a number of non-graphics calculations, such as solving differential equations, as well as for simulations and numerical analyses [1]. The GPU is, in general, very well suited for the computation of parallel problems and was also more recently employed as DSP for sound signal processing [2, 3]. In the area of sound simulations, the GPU was used to solve basic geometric room acoustics [4], as well as wave-based sound propagation using waveguide meshes [5]. Besides some physical differences, the propagation of sound and light share several similarities that make existing graphics rendering techniques exploitable to accommodate an acoustic energy propagation model.

The goal of this work is to build a foundation for ray-based sound simulations using an acoustic energy propagation model, and furthermore, to demonstrate its applicability and efficiency using modern graphics hardware and rendering techniques. We derive the acoustic rendering equations from global illumination models and radiometry used in computer graphics [6], and extend the existing model by time- and frequency dependencies. This paradigm is later employed in a GPU-based implementation to perform realtime sound simulations using ray-based techniques for the applications of room acoustics and personalized HRTF simulations. The audible spectrum is divided into 10 frequency bands, which are interpreted individually with respect to their wavelength and energy. The local energy contribution of each surface patch is evaluated separately per frequency band using functions of reflection, transmission/refraction, absorption and diffraction. Finally, the acoustic energy at the observers position is accumulated and filtered regarding direction and distance using HRTFs. The system allows us to simulate realtime interactive and dynamic environments with varying acoustic materials, but also to approximate individual HRTFs through ray-acoustic simulations.

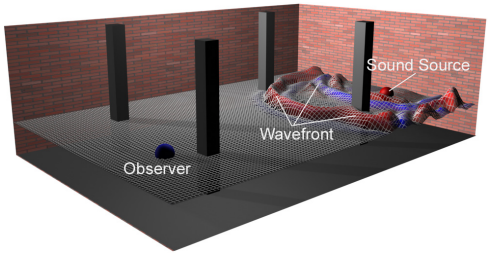
The paper is organized as follows: After this introduction, we review in Section 2 the existing approaches for sound simulations and compare their advantages and drawbacks. Section 3 follows up on the ray-based approach and develops a model for the propagation of acoustic energy in enclosures. This model studies the flow of acoustic energy from sound sources, its local interaction with objects and materials, as well as the measurement using a scene mounted listener. The following Section 4 maps the here developed concepts onto graphics primitives and rendering techniques, and discusses its implementation using modern programmable graphics hardware. Section 5 presents and discusses results using examples from room acoustic simulations and personalized HRTF measurements. The closing Section 6 summarizes the work and discusses several ideas for future improvements.

2. ACOUSTIC SIMULATION TECHNIQUES

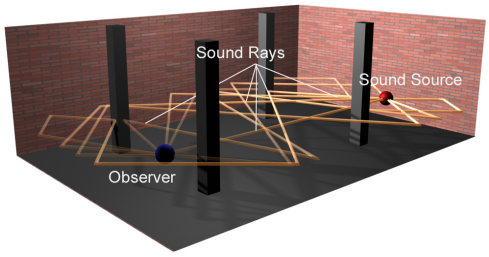
Auralization is defined as the simulation and reproduction of the acoustic properties describing a virtual scene, which has applications in many areas, including architectural design, sound and mu-

sic production and even audio-based computer games [7]. An accurate and efficient simulation is thereby still a difficult and computationally extensive task.

The most often employed approaches are waveguide meshes and raytracing techniques, see also Figure 1. Figure 1(a) displays here a visualization of the waveguide technique, a more physically correct wave-based sound propagation model, based on time-domain finite difference meshes. The acoustic energy, eg. pressure, is distributed along sampling points using difference equations. Figure 1(b) shows a visualization of the ray-based approach that approximates sound waves through particles and acoustic energy, and were raytracing techniques are used to determine the virtual soundfield. As both techniques have their own advantages and limitations, the wave-oriented techniques are usually employed for the lower frequency end, while the ray-based techniques are used for the middle and higher frequency parts.



(a) Wave-based Approach.



(b) Ray-based Approach.

Figure 1: Acoustic Simulation Techniques.

2.1. Wave-based Acoustics

Wave-based room acoustics is concerned with the numerically evaluation of the wave equation in order to simulate sound wave propagation. Often employed techniques are finite element methods (FEM) and 3D waveguide meshes (time-domain difference models) [8, 9]. The 1-dimensional waveguide technique is a numerical solution to the wave equation and was first applied to simulate string-based musical instruments [10]. The digital waveguide mesh is an extension of the 1D technique and constructed by bilinear delay lines that are arranged in a mesh-like structure [8]. Higher dimensions are built by scattering junctions that are connected to the delay lines and act as spatial and temporal sampling points. The equations that govern the rectilinear waveguide mesh are based on difference equations derived from the Helmholtz equation by discretizing time and space [11]. Depending on the mesh's resolution and the internodal sampling distance, the simulations can be rather expensive. Due to advances in computing power, realtime wave-based room acoustics is feasible for smaller meshes.

Although, the simulations using waveguide meshes are very accurate, there are some drawbacks as well. The two major problems are a direction dependent dispersion error, and a finite mesh resolution to model a more complex boundary behavior [8]. Several approaches have been discussed to overcome these limitations and include higher tesselated meshes, different mesh topologies and frequency warping techniques [12, 13]. Additionally, the sampling frequency of the rooms impulse response needs to be oversampled, with previous research showing that a typical waveguide mesh gives a valid bandwidth only as far as $f_{update}/4$ [8]. Therefore, this technique is only practical to the very lower frequency end. However, recent research has shown that waveguide meshes can easily and efficiently be implemented using graphics hardware. Combined with a new sampling lattice, the performance increase was measured by a factor of 25, and even more for finer mesh resolutions [5].

2.2. Geometric Acoustics

Geometric acoustics is based on optical fundamentals and light propagation and approximates sound waves through particles moving along directional rays [14, 15]. These rays are traced through a virtual scene, starting at the sound source and towards a listeners position, at which the accumulated energy is later evaluated. As sound waves are now simplified as rays, wave phenomena and differences in wavelength are usually discarded and ignored. This method is therefore only applicable to frequencies whose wavelength are much shorter than the dimensions of the enclosure, or any object within, refer also to [16, 17].

Several articles have been published over the last years, which discuss the realtime possibilities of ray-acoustic sound simulations [18, 19]. The majority of implementations, however, employs raytracing only to determine specular reflections using ray/beam-tracing approaches and uses conventional 3D sound APIs for spatialization and sound rendering [14, 15, 4]. As raytracing is a long known area of research in computer graphics, several improvements and advancements to the original approach have been proposed, and were partially applied to ray-acoustics as well. Savioja et.al. have designed the DIVA auralization system based on a ray-acoustics approach, to examine modeling techniques for virtual acoustics, as well as for physically-based auralizations [20, 21].

Some of the more recent geometric acoustic implementations already utilize computer graphics hardware to increase the simulations efficiency. Jedrzejewski uses the GPU for simple 2D geometric room acoustics using rays and specular reflections [4], while Kapralos and Deines employ a particle-based system to adopt the phonon mapping technique towards a *phonon tracing* approach [22, 23, 24]. Although, this technique allows an accurate modeling of acoustic materials and sound propagation, it only permits static and non-changing environments. Interesting, from the perspective of a complete GPU-based sound simulation and rendering approach, is also the work by Gallo and Whalen [3, 2], who employ the GPU as DSP for sound signal filtering and synthesis.

3. ACOUSTIC ENERGY PROPAGATION

Sound is the propagation of mechanical energy in the form of pressure variations and can be described by attributes such as frequency, wavelength, speed of propagation etc. Light on the other hand is an electromagnetic radiation, which is described by similar, however, largely different quantities. The propagation of light

f_j	f_{range_j} (Hz)	f_{center_j} (Hz)	λ_{center_j} (m)
f_0	22 – 44	31.5	10.88
f_1	44 – 88	63	5.44
f_2	88 – 177	125	2.74
f_3	177 – 354	250	1.37
f_4	354 – 707	500	0.68
f_5	707 – 1,414	1,000	0.343
f_6	1,414 – 2,828	2,000	0.172
f_7	2,828 – 5,657	4,000	0.086
f_8	5,657 – 11,314	8,000	0.043
f_9	11,314 – 22,627	16,000	0.021

Table 1: Frequency Bands f_j.

energy and its interaction with objects can be measured and described by using techniques of radiometry, from which global illumination models used in computer graphics are derived [6]. The concepts of radiometry, along its properties and equations, can be mapped to the propagation of acoustic energy as well. This assumes that the propagation of sound waves can be simplified to a ray-based approach by largely neglecting characteristics such as wavelength, diffraction and interference. For middle- and higher frequencies, and depending on the rooms and enclosed objects size, this assumption is true to a certain degree. Especially at the lower frequency end wave-based effects become such prominent that they prevail. Therefore, the here discussed model also addresses these issues and incorporates the wavelength to approximate diffraction and interference effects. The following sections discuss the theories behind, and extend the concepts of radiometry towards a ray/energy-based acoustic propagation model suitable for sound wave simulations.

3.1. Emission and Radiation

In order to study and describe the propagation of sound waves using raytracing techniques, an adequate propagation model that incorporates time- and frequency dependencies needs to be defined. This can be realized in analogy to the physics of light transportation and global illumination models [6], which now have to be extended and adopted towards acoustic properties and an acoustic energy propagation [25].

Whereas the wavelength of the visible spectrum ranges only between 380 nm to 780 nm, the wavelength in acoustics spreads from 17 mm at 20 kHz up to 17 m at a frequency of 20 Hz. The frequencies in the audible spectrum are classified and described by frequency bands (octaves) according to human psychoacoustics. In the following sections f_j describes a certain frequency band, with j being the index number and $j+1$ the next higher octave. Table 1 provides an overview of the different frequency bands, along their index number, frequency range f_{range_j}, center frequency f_{center_j} and center wavelength λ_{center_j}. The audible spectrum $A_{spectrum}$ is therefore defined as the sum of these 10 frequency bands:

$$A_{spectrum} = A_s = \sum_{j=0}^{9} f_j. \tag{1}$$

Similar to light, acoustic energy can be described as the amount of pressure variations per unit volume and time, or more accurately, by the changes in velocity of air particles contained in a volume element per unit time. The quantity for describing and measuring acoustic energy is radiant power Φ, or flux, and measured in $Watt$ or $Joule/sec$ [6]. The intensity is thereby described as the amount of acoustic energy flowing from/to/through a surface element per unit time:

$$I(t) = \frac{d\Phi}{dA}dt. \tag{2}$$

The transfer of acoustic energy using a participating media (air) is characterized by the energy transport theory. The energy density in the medium of propagation is hereby the sum of the kinetic and potential energy per unit volume dV and time $E(t) = E_{kin}(t) + E_{pot}(t)$ [25]. The kinetic energy density is defined as the pressure of a sound wave as:

$$E_{kin}(t) = \frac{1}{2}\frac{Mv^2}{V_0}dt = \frac{1}{2}\rho_0 v^2 dt, \tag{3}$$

with v being the average velocity of air particles, ρ_0 the average media density and $\frac{M}{V_0}$ its mass per unit volume V_0. The potential energy density can be derived from the gas law as:

$$E_{pot}(t) = \frac{\int pdp}{c^2\rho_0}dt = \frac{1}{2}\frac{p^2}{c^2\rho_0}dt, \tag{4}$$

with p as the pressure of the sound wave and c as the speed of sound in this medium, and therefore defines the total amount of acoustic energy density [25] as:

$$E(t) = E_{kin}(t) + E_{pot}(t) = \frac{1}{2}(\rho_0 v^2 + \frac{p^2}{c^2\rho_0})dt. \tag{5}$$

Equation 5 is valid at any position and time within the virtual auditory environment and serves as basis to describe an acoustic energy propagation model. In order to quantitatively measure flux per unit projected surface area and per unit angle, radiance is introduced with:

$$L(x,\Theta) = \frac{d^2\Phi}{d\omega dA cos\theta}, \tag{6}$$

which varies with position x and the ray's direction Θ. By incorporating the wavelength λ_j of the frequency bands used (ref. Table 1), Equation 6 is redefined to:

$$L(x,\Theta,f_j) = \int_{A_s} L(x,\Theta,f_j)d\lambda. \tag{7}$$

The acoustic energy interacting with a surface element can be further differentiated in incident E_i (incoming) and exitant E_e (outgoing) energy, and is also measured in $Watt/m^2$:

$$E_i = \frac{d\Phi}{dA}, E_e = kE_i. \tag{8}$$

The scalar k is herby defined over $[0,1]$ and describes the reflectivity of the surface with $E_{surface} = E_i - E_e$ and is affected by the surface material definition. Using a lossless participating media, the exitant radiance at one point $L(x_1 \rightarrow \Theta)$ is exactly the same as the incident radiance at another point receiving this amount of energy $L(x_2 \leftarrow \Theta)$ [6]. Using a density function and volume elements, $p(x)dV$ defines the physical number of *sound particles* carrying an *acoustic energy quant*. If moved in time dt across a differential surface area dA, and by using the direction ω and speed of propagation c; $N = p(x,\omega,f_j)cdtdAcos\theta d\omega d\lambda$

describes the number of particles flowing through this surface element. The radiance per unit volume is accordingly redefined to:

$$L(x, \Theta, f_j) = \int_{A_s} \int p(x, \omega, f_j) h \frac{c}{\lambda_j} d\lambda. \quad (9)$$

An energy/sound source emits acoustic energy that is propagated through and by the participating media. The energy radiates through an emittance pattern, which can be homogenous in any direction, eg. spherically, or direction dependent, such as a cone. As with light, also acoustic energy attenuates with distance using the familiar inverse square law. Furthermore, atmospheric absorption occurs, at which certain frequencies are absorbed by the propagating media. However, this factor is very small and can safely be ignored for smaller enclosures, but becomes more prominent with increasing distances.

An observer, or listener, can be placed anywhere within the scene to *record* the acoustic energy present at this location. The listener does not interfere or participate in the energy propagation, but, if required, such as for binaural listening, an additional geometry can be placed nearby to simulate head-shadowing effects. The incoming rays are then weighted and filtered using HRTFs regarding the ray's direction and delay.

3.2. Local acoustic Energy Exchange

The most interesting part in a ray-based acoustic simulation is the interaction and exchange of acoustic energy with objects and surface elements. Depending on the objects size and the acoustic material parameters specified, some of the incoming energy might get absorbed, reflected, refracted or transmitted, with the total amount of energy according to Equation 8 being constant.

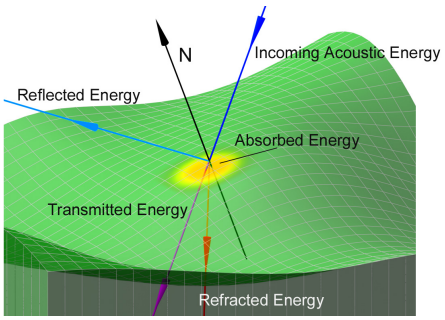


Figure 2: Local acoustic Energy Exchange.

Figure 2 shows a schematic of the local acoustic energy exchange. The four effects of absorption, reflection, refraction and transmission are described in more detail in the remainder of this section. Every ray that is cast into the scene contains, depending on the sound source emittance of course, the energy of all frequency bands. The energy contribution of each ray is evaluated at the point of intersection with the surface patch using the ray's length, as well as the surface material properties defined.

Some of the incident acoustic energy is thereby usually absorbed, converted into heat and dissipated back into the system. The absorption is frequency dependent and characterized by a frequency band coefficient α_{f_j}:

$$L_{e_{absorbed}}(x \leftarrow \Theta) = \sum_{j=0}^{9} E_{i_j} \alpha_{f_j}. \quad (10)$$

Transmission is defined as the energy that passes through an object. We redefine this term to describe the frequency-weighted amount of energy that passes through an object *unaltered* and without refraction. In acoustics, objects smaller than the wavelength of an incoming sound wave do not interfere, instead the wave simply diffracts around the object and continues unchanged. An according frequency dependent modeling of energy transmission can be realized using an objects bounding box or sphere that simply transmits all acoustic energy whose wavelength is equal or above the objects size:

$$L_{e_{transmitted}}(x \rightarrow (\pi + \Theta)) = \sum_{j=0}^{9} E_{i_j} \tau_{f_j}. \quad (11)$$

Here $L_{e_{transmitted}}(x \rightarrow (\pi + \Theta))$ describes the amount of exitant energy per ray for all bands, which simply pass along the direction opposite to the incoming ray, i.e. the ray's original direction. The term τ_{f_j} is used for a finer modeling and a frequency-weighting of the transmission effects.

Reflection and diffuse scattering are probably the two most important qualities in acoustic raytracing and can be very well described using bidirectional reflection distribution functions (BRDF) [6]. A BRDF is defined for a point x as the ratio of the differential radiance reflected in an exitant direction Θ_e and the differential irradiance incident through an incoming angle Θ_i:

$$brdf_{reflected}(x, \Theta_i \rightarrow \Theta_e) = \frac{dL(x \rightarrow \Theta_e)}{dE(x \leftarrow \Theta_i)}. \quad (12)$$

The BRDF is frequency dependent, but direction independent, eg. $f_r(x, \Theta_i \rightarrow \Theta_e) = f_r(x, \Theta_e \rightarrow \Theta_i)$ [6, 26]. Diffuse scattering uniformly reflects the incoming acoustic energy in all directions. In acoustics, this behavior is largely influenced by the surface roughness, which can be used to determine a specular reflection coefficient that describes the ratio between specular and diffuse reflections. Using a complete diffuse scattering, the radiance is independent from the angle of exitance and the BRDF defined as:

$$brdf_{reflected}(x, \Theta_i \leftrightarrow \Theta_e) = \frac{\rho_{diffuse}}{\pi}, \quad (13)$$

in which the reflectance $\rho_{diffuse}$ represents the fraction of incident energy reflected at the surface. Pure specular reflection on the other hand diverts all incident energy in only one direction R, which can be simply computed using the law of reflection and the surface normal N: $2(N(\pi + \Theta_e))N - (\pi + \Theta_e)$. A frequency dependent BRDF for acoustic raytracing can be modeled through:

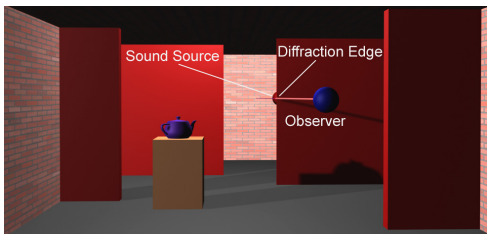
$$L_{e_{reflected}}(x \leftarrow \Theta_i) = \sum_{j=0}^{9} E_{i_j} \upsilon_{f_j}, \quad (14)$$

in which υ_{f_j} is a weighting factor per frequency band f_j. The majority of materials, however, exhibit a sort of *glossy* surface, a combination of specular reflection and diffuse scattering.

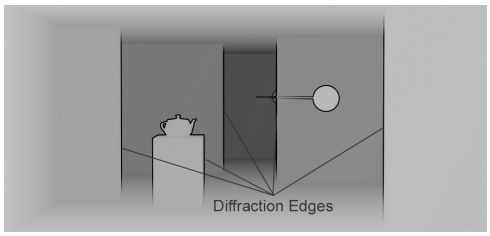
Refraction occurs at the crossing of two different isotropic media and can be computed similar to the reflection term in Equation 12, except that the outgoing angle Φ of the refracted ray is determined using Snell's Law: $sin\Phi = \frac{\eta_2}{\eta_2}$. Here η_1 and η_2 are the refraction indices of their respective media. A frequency band weighted refraction can be defined similar to Equation 14 by using ν_{f_j} as weighting coefficient per frequency band.

3.3. Diffraction and Interference

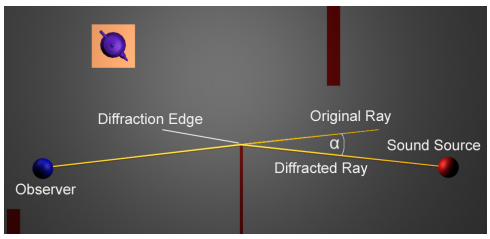
Edge diffraction and interference are acoustic phenomena that can be modeled accurately using wave-based techniques, but do not fit well into the concept of ray-acoustic simulations. However, both are very important and prevail especially in the lower frequency ranges. Therefore, and in order to obtain a more *realistic* simulation, these effects have to be included, or at least approximated. Generally, this is done by combining wave-based and ray-based approaches and by choosing a certain threshold as boundary frequency. But to a certain degree, these effects can also be approximated within ray-acoustics.



(a) Scene Rendering as seen from Listener's Position.



(b) Combined Depth/Edge Map.



(c) Top View with original and diffracted Ray.

Figure 3: Ray Acoustic Diffraction Simulation.

Sound waves with larger wavelength simply bend around edges, such as if an additional sound source was placed at the diffraction edge. Diffraction effects are in ray/energy acoustics simply modeled through ray-bending, according to the ray's length and its associated frequency band f_j. As diffraction is dependent on the objects size and the ray's wavelength, the amount of energy that is diffracted is determined individually per frequency band f_j. The maximum possible diffraction angle was herby determined experimentally using a wave-based sound propagation system [5]. Figure 3 visualizes the concept of the implemented diffraction system. It shows a virtual scene from the listener's perspective (Figure 3(a)), the constructed edge map (Figure 3(b)) and the by angle α diffracted ray from a listener to a sound source (Figure 3(c)). For each edge in Figure 3(b), additional rays are cast into the scene for diffraction simulation.

Interference describes the superposition of two or more sound waves and the resulting changes in amplitude. Using a ray-acoustic sound simulation, interference effects can only be approximated roughly using the ray's length and the center wavelength λ_{center_j} of the current frequency band f_j. By using an additional scalar associated with each ray, also the modeling of phase-preserving and phase-reversing reflections are possible. The next section focusses after these theoretical discussions on the implementation of the here described acoustic energy propagation model using efficient computer graphics hardware.

4. GRAPHICS-BASED RAY ACOUSTIC SIMULATIONS

While the last section discussed the propagation of acoustic energy and its interaction with objects and materials, this section maps the there developed concepts onto computer graphics primitives and rendering equations. The presented framework implements a ray-based acoustic simulation system that exploits modern computer graphics hardware. The system is designed along current GPU-based raytracing systems [27, 28, 29], which were extended towards the acoustic energy propagation model as discussed in the last section. The advantages and applicabilities of such an implementation can be summarized as:

- Efficient ray-based acoustic simulation system that incorporates wave phenomena,
- Realtime implementation that exploits graphics hardware,
- Built-in visualization of sound wave propagation,
- An eclectic modeling and design of acoustic materials, with
- Applications for impulse response measurements and general room acoustics, as well as to
- Approximate individualized HRIRs.

The system takes any 3D polygonal mesh as input, which is pre-processed into a more efficient accessible structure. It allows an interactive sound simulation for meshes of up to 15,000 polygons. Using a short pulse as sound signal, room impulse response (RIR), as well as head-related impulse response (HRIR) measurements are possible. Alternatively, a monaural sound file can be used as input signal, resulting in a spatialized binaural representation with the virtual rooms imprint. The sound source/listener positions, as well as the acoustic material definitions can be changed and adjusted interactively. All sound signal processing, including HRTF convolution and delay filtering, is realized using fragment shaders onboard the graphics hardware.

4.1. Auralization Pipeline

The auralization pipeline employed in our system stretches over the CPU and GPU systems, but the majority of computations is carried out in graphics hardware. Figure 4 shows an overview of the pipeline, along its partition in CPU and GPU related tasks. As initialization, 3D scene data, as well as sounds and frequency band decomposed HRTFs are loaded into texture memory. The sound data is also decomposed into 10 bands and assigned a position and emittance pattern within the virtual room. Rays are now cast into the scene starting at the listeners position, and the per frequency band received acoustic energy is accumulated and stored within so called cubemaps. This cubemap is later evaluated and the sound data is filtered and delayed using HRTFs according to their position and the ray's length. The binaural mixdown is performed

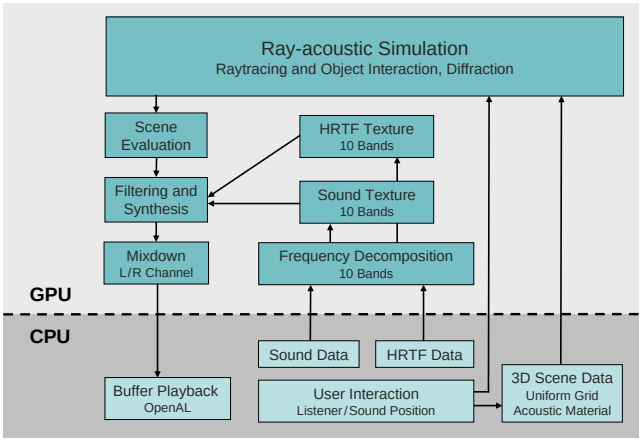


Figure 4: Auralization Pipeline.

using a two-channel floating point texture, which is streamed back to the CPU and fills a native OpenAL stereo buffer for sound playback.

4.1.1. Uniform Grid Structure

In a pre-processing step, the 3D scene is converted into a uniform grid structure that subdivides 3D space and groups neighboring triangles together in a voxel-based topology. These voxels are of uniform size and axis aligned. This space subdivision is necessary in order to efficiently determine ray/object intersections, as now only the triangles grouped in one voxel element have to be tested [27, 28, 29]. Care has to be taken in defining the voxel's size, as with very detailed objects the number of polygons can easily exceed the number of possible shader instructions.

4.1.2. Frequency Decomposition and Synthesis

A frequency-based acoustic raytracing has many advantages, as now some of the wave-based propagation effects can be approximated, as well as it allows a more realistic frequency-dependent definition of acoustic materials. Currently we employ 10 frequency bands, grouped into octaves as known from psychoacoustics, see Table 1. For the frequency decomposition of sound data and HRTFs, we employ a time-based convolution using windowed sinc filters, with their cutoff frequencies specified as the bands respective border frequencies. These 10 bands are loaded as floating point textures into graphics hardware. To remain data precision, we currently employ 3 RGBA textures to hold the sound data, although, using data compression, two should be sufficient for 16 bit sound data. Ray/object interactions are evaluated per frequency band and the contributions from each ray are accumulated and also stored individually. The final auralization is a binaural sound signal that is generated by filtering the original sound texture using HRTFs according to the simulations result.

4.2. Acoustic Raytracing and Diffraction Simulation

The authoring of 3D scenes can be conveniently performed using 3D Studio MAX, where a custom-built plugin is used to assign acoustic material definitions to each object. This acoustic material defines the wavelength specific energy exchanges for each surface patch. Although, all materials are assigned per vertex, no interpolation of neighboring material attributes is performed yet.

The raycasting and acoustic energy accumulation is carried out using so called cubemaps. One cubemap is hereby centered around the observers position and a ray is cast into the scene per cubemap texel. Figure 5 shows a visualization of this cubemap raycasting approach. Each ray cast is traced through the virtual scene and its acoustic energy accumulated and stored per frequency band. At points of ray/object intersection, the local surface acoustic energy exchange is evaluated according to Section 3.2. Newly generated rays from refraction, transmission and/or reflection are further traced, until their possible energy contribution falls below a certain threshold ϵ. The cubemap not only stores all incoming acoustic energy per frequency band, but also the ray's direction and length. This information is later used for the final binaural sound signal synthesis.

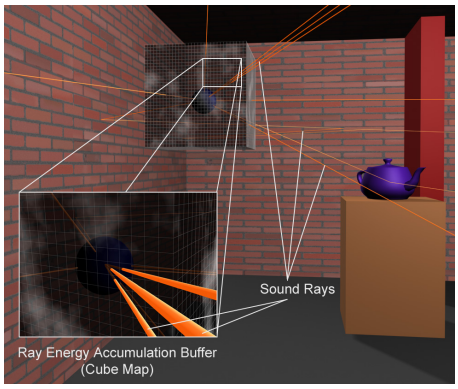


Figure 5: Ray Energy Accumulation Buffer.

4.2.1. Diffraction Simulation

The ray acoustic simulation also incorporate diffraction effects on edges and object borders. To find possible diffraction locations, a depth/edge map is employed, which highlights these edges, see also Figure 3(b). These maps are created by using the scenes depth buffer and an image-based edge detection algorithm. If a ray is cast close to a diffraction edge, the ray is bend according to the diffraction of sound waves [16, 17]. Here the ray's energy is attenuated, depending on the angle and the ray's wavelength. Another wave-based phenomena is interference, which can be roughly approximated by using the ray's length and the frequency bands center wavelength λ_{center_j}, see Table 1. Although, this is a very simple approximation, it would also allow the modeling of phase-reversing and -preserving boundary reflections, as well as to use this information for interference effects of the same and/or different frequency bands.

4.3. Implementation

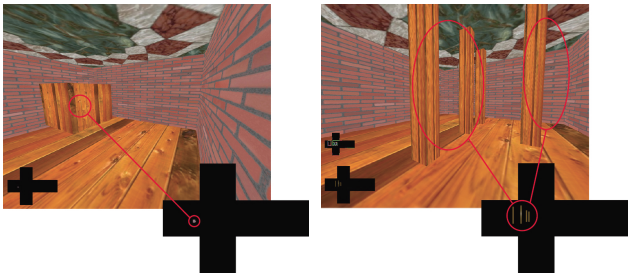
Today's graphics hardware, and especially the new generation with its unified shader architecture, can be seen as powerful parallel processing machines, which can very efficiently execute small programs - so called shaders - in parallel. Shaders are freely programmable using high level shading languages such as GLSL and Cg. As graphics applications typically require the processing of huge amounts of data, graphics hardware has been optimized to support

this with a highly parallel design. Combined with a fast and optimized memory architecture for accessing and storing the data, this makes this hardware very interesting for any computationally intensive and parallelizable task.

All convolutions and sound synthesis are carried out using fragment shaders on graphics hardware, with a single shader for each task. The data, eg. sounds, geometry and material definitions are stored within textures and accessed during the rendering task from within the shaders. The results of the simulation are again stored as textures, from which they are read back to the CPU for sound playback.

5. RESULTS AND DISCUSSION

This section discusses some results of the ray-acoustics simulation system. The implementation is based on *n*vidia type graphics hardware and uses Cg as shading language. The current experiments were performed with three different graphics hardware generations, showing that only the newest one (GeForce8800GTX) was also able to additionally perform a realtime auralization of the results besides the sound simulation. Frame rates of up to 25 fps could be achieved using a detailed model of a living room (1,500 polygons) including a binaural auralization of the scene, ref. Figure 7(b).



(a) Transmission and Refraction. (b) Diffraction around Pillars.

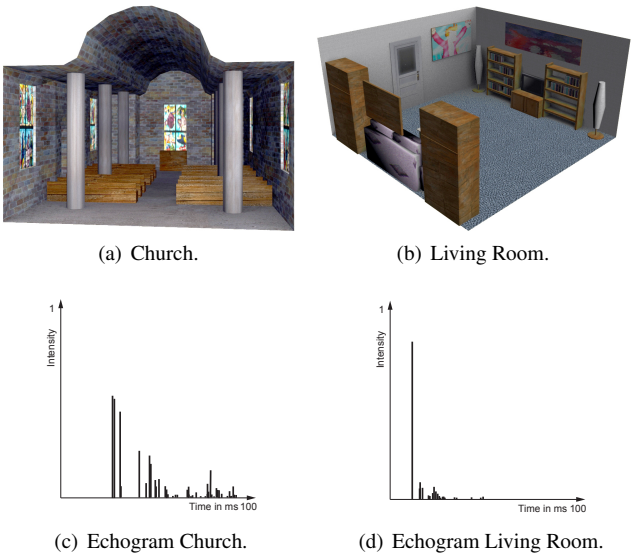
Figure 6: Evaluation of Sound Propagation Effects.

Figure 6 shows two visualizations of sound propagation effects. Here Figure 6(a) displays the transmission and refraction parts of the simulation, whereas Figure 6(b) shows diffraction effects of several pillars. In both cases the sound source is hidden and the simulation results are visible in the unfolded cubemaps below. Both cubemaps show a red/braun shifting of the color, denoting a stronger transmission/diffraction in the lower frequencies.

5.1. Example 1: Room Acoustics

The first example shows two different rooms along their echograms. Figure 7(a) displays thereby a small church, while Figure 7(b) shows an average living room. The echogram of the church, ref. Figure 7(c) shows strong and late echoes, while the echogram in Figure 7(d) shows that nearly all acoustic energy, except the direct line, was absorbed by walls and furniture. Both echograms clearly visualize the rooms acoustic properties. Each room has been modeled using 3D Studio MAX, in which for each surface a different materials has been specified. The properties for the acoustic material definitions were taken from the CARA database[1]. The

[1] `http://www.cara.de`



(a) Church. (b) Living Room.

(c) Echogram Church. (d) Echogram Living Room.

Figure 7: Room Acoustic Example Scenarios.

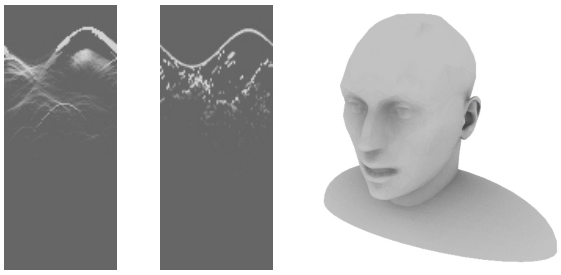
echograms show the early reflections, as well as late reverberation and diffraction effects.

5.2. Example 2: Personalized HRTF

The second example shows an HRIR simulation of the horizontal plane using our ray-acoustics approach. The simulation was performed using 72 sound sources, each 1.2 m apart from the head at a 5 degree interval. Although, the simulation does not exhibit all effects of a regular measured HRIR, it shows the most prominent features. The simulation was performed using a 3D model of the KEMAR mannequin. Figure 8 shows two different simulation results, along the original 3D model used. Here Figure 8(a) displays an HRIR simulation of the system from [30], while Figure 8(b) show the results of the here presented ray-acoustics system. Thereby roughly 18 million rays were traced per sound source, resulting in a simulation time of 22 seconds per sound source. Although the most important features are clearly present, several effects are still missing. This is partially due to the fact that we only consider one diffraction per ray. Also, a more detailed fine tuning of parameters along the material definitions for the head, torso and ear will yield better results. The goal is to combine an individualized HRTF simulation with room acoustics, to yield a realtime personalized binaural room simulation. Better results with geometric models have been achieved by [31], but, however, also with a much longer simulation time.

6. CONCLUSIONS AND FUTURE WORK

We have presented a realtime graphics-based implementation of a ray acoustic simulation system that is based on an acoustic energy propagation model. This underlying model is founded on sound propagation, as well as global illumination models, and the ray/energy approach used therefore valid and its implementation using graphics hardware and techniques viable. The current results clearly show the possibilities of this system and motivate a further research in this area.



(a) HRIR 1. (b) HRIR 2. (c) KEMAR Model.

Figure 8: HRIR Simulation and 3D Model.

The current implementation already exhibits good and very promising results, yet some ideas are left for future improvements. One extension would be an enhanced space partitioning structure, such as kD-Trees that allow a non-uniform subdivision of 3D space. Future work also includes more and finer partitioned frequency bands for a more accurate studying and modeling of wave-based propagation effects. Another beneficial extension would be a higher incorporation of radiosity techniques, although one has to be careful to not impede here with realtime simulations and dynamic environments. Additionally, ambisonics and their implementation using spherical harmonics in realtime computer graphics might be an interesting path to explore.

7. ACKNOWLEDGEMENTS

The authors would like to thank Yuvi Kahana, Brian FG Katz, as well as Richard O. Duda for providing a polygonal mesh of the KEMAR head for the HRIR simulations.

8. REFERENCES

[1] John D. Owens, David Luebke, Naga Govindaraju, Mark Harris, Jens Krüger, Aaron E. Lefohn, and Timothy J. Purcell, "A survey of general-purpose computation on graphics hardware," in *Eurographics 2005, State of the Art Reports*, Aug. 2005, pp. 21–51.

[2] Sean Whalen, "Audio and the Graphics Processing Unit," Tech. Rep., 2005, `http://www.node99.org/projects/gpuaudio/`.

[3] Emmanuel Gallo and Nicolas Tsingos, "Efficient 3D Audio Processing with the GPU," in *GP2, ACM Workshop on General Purpose Computing on Graphics Processors*, 2004.

[4] Marcin Jedrzejewski, "Computation of Room Acoustics Using Programmable Video Hardware," in *International Conference on Computer Vision and Graphics*, 2004.

[5] Niklas Röber, Martin Spindler, and Maic Masuch, "Waveguide-based Room Acoustics through Graphics Hardware," in *ICMC*, 2006.

[6] Philip Dutre, Philippe Bekaert, and Kavita Bala, *Advanced Global Illumination*, AK Peters Ltd., 2003.

[7] Niklas Röber and Maic Masuch, "Leaving the Screen: New Perspectives in Audio-only Gaming," in *Proceedings of 11th ICAD*, 2005, Limerick, Ireland.

[8] S. VanDuyne and J.O. Smith, "Physical Modelling of the 2-D digital Waveguide Mesh," in *Int. Computer Music Conference*, Tokyo, Japan, 1993, pp. 40–47.

[9] Y. Kahana, P.A. Nelson, M. Petyt, and S. Choi, "Numerical Modelling of the Transfer Functions of a Dummy-Head and of the external Ear," in *Audio Engineering Society, 16th International Conference*, Rovaneimi, 1999, pp. 330–345.

[10] Julius O. Smith, "Physical modelling using digital Waveguides," *Computer Music Journal*, vol. 16, no. 4, pp. 75–87, 1992.

[11] Lauri Savioja and Tapio Lokki, "Digital Waveguide Mesh for Room Acoustic Modelling," in *ACM SIGGRAPH Campfire: Acoustic Rendering for Virtual Environments*, Utah, USA, 2001.

[12] Mark J. Beeson and Damian T. Murphy, "Roomweaver: A digital Waveguide Mesh based Room Acoustics Research Tool," in *7th Int. Conference on Digital Audio Effects*, Italy, 2004, pp. 268–273.

[13] F. Fontana and D. Rocchesso, "Signal-Theoretic Characterization of Waveguide Mesh Geometries for Models of Two-Dimensional Wave Propagation in Elastic Media," *IEEE Transactions on Speech and Audio Processing*, vol. 9, no. 2, pp. 152–161, Februar 2001.

[14] Wolfgang Mueller and Frank Ullmann, "A Scalable System for 3D Audio Ray Tracing," in *Proceedings of the IEEE Int. Conference on Multimedia Computing and Systems*, 1999, pp. 819–823.

[15] Thomas Neumann, "Eine Geometrie-Engine zur Berechnung von 3D-Sound mit Raumakustik-Effekten in Echtzeit," M.S. thesis, Tchnische Universität Braunschweig, Institut für Compuergraphik, 2004.

[16] F. Alton Everest, *The Master Handbook of Acoustics*, vol. 4th edition, McGraw-Hill/TAB Electronics, 2001.

[17] Heinrich Kuttruff, *Room Acoustics*, Spon Press, London, 2000.

[18] Thomas Funkhouser, Nicolas Tsingos, Ingrid Carlbom, Gary Elko, Mohan Sondhi, and James West, "Modeling Sound Reflection and Diffraction in Architectural Environments with Beam Tracing," *Forum Acusticum*, 2002.

[19] Emmanuel Tsingos, Nicolas Gallo and George Drettakis, "Perceptual Audio Rendering of Complex Virtual Environments," in *Proceedings of ACM SIGGRAPH*, 2004.

[20] Lauri Savioja, Tapio Lokki, and Jyri Huopaniemi, "Auralization applying the Parametric Room Acoustic Modelling Technique - The DIVA Auralization System," in *Proceedings of ICAD*, Japan, 2002.

[21] Lauri Savioja, *Modelling Techniques for Virtual Acoustics*, Ph.D. thesis, Helsinki University of Technology, Finnland, 2000.

[22] B. Kapralos, M. Jenkin, and E. Milios, "Sonel Mapping: Acoustic Modeling utilizing an acoustic Version of Photon Mapping," in *IEEE International Workshop on Haptic Audio Visual Environments and their Applications*, 2004.

[23] Martin Bertram, Eduard Deines, Jan Mohring, Jevgenij Jegorovs, and Hans Hagen, "Phonon Tracing for Auralization and Visualization of Sound," in *IEEE Visualization*, Minneapolis, USA, 2005.

[24] Eduard Deines, Martin Bertram, Jan Mohring, Jevgenij Jegorovs, Frank Michel, Hans Hagen, and Gregory M. Nielson, "Comparative Visualization for Wave-based and Geometric Acoustics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, September/October 2006.

[25] Leo L. Beranek, *Acoustics*, Amer Inst of Physics, 1986.

[26] Zhiyun Li, Ramani Duraiswami, and Nail A. Gumerov, "Capture and Recreation of higher Order 3D Sound Fields via Reciprocity," in *ICAD*, 200f.

[27] Timothy J. Purcell, Ian Buck, William R. Mark, and Pat Hanrahan, "Ray Tracing on Programmable Graphics Hardware," in *Proceedings of ACM SIGGRAPH*, 2002, pp. 703–712.

[28] Timothy J. Purcell, *Ray Tracing on a Stream Processor*, Ph.D. thesis, Stanford University, March 2004.

[29] Gabriel Moreno-Fortuny, "Distributed Stream Processing Ray Tracer," Tech. Rep., University of Waterloo, `http://www.cgl.uwaterloo.ca/~gmoreno/streamray.html` 2004.

[30] Niklas Röber, Sven Andres, and Maic Masuch, "HRTF Simulations through acoustic Raytracing," Tech. Rep., Fakultät für Informatik, Otto-von-Guericke Universität Magdeburg, 2006.

[31] V.R. Algazi, R.O. Duda, R. Duraiswami, N.A. Gumerov, and Z. Tang, "Approximating the Head-related Transfer Function using simple Geometric Models of the Head and Torso," *Journal of the Acoustical Society of America*, , no. 5, pp. 2053–2064, November 2002.

IMPLEMENTING DIGITAL AUDIO EFFECTS USING A HARDWARE/SOFTWARE CO-DESIGN APPROACH

Markus Pfaff, David Malzner, Johannes Seifert, Johannes Traxler, Horst Weber, Gerhard Wiendl

FH-OÖ/Hagenberg, Dept. HSSE
Softwarepark 11, A-4232 Hagenberg/Austria

ABSTRACT

Digital realtime audio effects as of today are realized in software in almost all cases. The hardware platforms used for this purpose reach from multi purpose processors like the Intel Pentium class over embedded processors (e.g. the ARM family) to specialized DSP.

The upcoming technology of complete systems on a single programmable chip contrasts such a software centric solution, because it combines software and hardware via some co-design methodology and makes for a promising alternative for the future of realtime audio. Such systems are able to combine the vast amount of computing power provided by dedicated hardware with the flexibility offered by software in a way the designer is free to influence.

While the main realization vehicles for these systems – FPGAs – were already promising but unfortunately offered limited possibilities a decade ago [1] they have made rapid progress over the years being one of the product classes that drive the silicon technology of tomorrow.

We describe an example for such a realtime digital effects system which was developed using a hardware/software co-design method. While digital realtime audio processing takes place in low latency dedicated hardware units the control and routing of audio streams is done by software running on a 32 bit NIOS II softcore processor. Implementation of the hardware units is done using a DSP centric methodology for raising the abstraction level of VHDL descriptions while still making use of standard of the shelf FPGA synthesis tools. The physical implementation of the complete system uses a rapid prototyping board tailored for communications and audio applications based on an Altera Cyclone II FPGA.

1. INTRODUCTION

Software running on a DSP or a common CPU is the prevalent vehicle of digital real-time audio effects implementation today. Realization of such effects in dedicated hardware has some appealing advantages especially in low latency and high reliability applications [2]. Little flexibility and a much more complicated design process than software does offer are the other side of the coin. These severe drawbacks have prevented dedicated hardware design from gaining ground in the digital audio effects realm at least in its consumer and semiprofessional floors.

Using the arithmetic package described in [3] which provides a fractional data type for fixed point digital data processing the abstraction level in terms of data handling and arithmetic expressions raised a lot over what is possible using the integer types typically encountered in such descriptions: signed and unsigned. The work described in [4] gave proof of concept for the general usefulness of the fractional package for digital signal processing as it is done in applications typically found in the communications industry. Such applications seldom make use of the large amount of dynamic parameters that many audio applications demand. Flexibility of a description simply is no issue in this case.

The desire to broaden the application area of our approach led to the decision to implement effects from the audio domain in hardware using a rapid prototyping board. The project DAFX [5] was launched at the University of Applied Sciences of Upper Austria at Hagenberg in October 2006 which aimed the evaluation the feasibility of a combined hardware/software approach for targeting the audio effects application field. Parameterization of the hardware audio effect modules is done through software running on a 32 bit softcore processor which is implemented together with the effects on an FPGA as a re-programmable System-on-Chip. Hardware and software subsystems together built a complete hardware/software co-design. The software controlled processor core also controls the data streams connecting the different hardware effect units.

We will point out advantages as well as disadvantages found while realizing a digital audio system this way. We start by describing the rapid prototyping system used as the realization platform. The second part of the paper deals with the used components and the basic setup for effect development followed by the description of the implementation and design funcional simulation based verification of effects.

2. RAPID PROTOTYPING BOARD SANDBOXX

The platform chosen for the realization of the system described in this paper is the rapid prototyping board *Sand-*

boxX which has been developed at the University of Applied Sciences of Upper Austria at Hagenberg and is used for educational and research purpose.

The board shown in Fig. 2 is built around an *Altera Cyclone II* FPGA (*EP2C35F*). Peripherals (see Fig. 1) of the FPGA are 16 megabytes of SDRAM, a *Texas Instruments* audio codec (*TLV320AIC23B*), a MIDI Interface, a PS/2 interface and a programmable clock IC (*ICS307*). Further hardware units such as a PCI Bus connector are available on the board, but were not used in the project. The board powers up with a default clock frequency of 25 MHz. Because the system uses a clock frequency of 48 MHz, we needed to reprogram the clock frequency. This is done on startup by software on the NIOS II processor. For configuration an SPI interface is used.

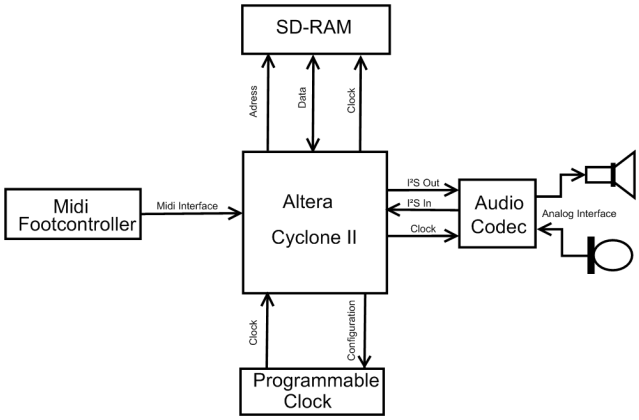


Figure 1: Hardware for the audio effects

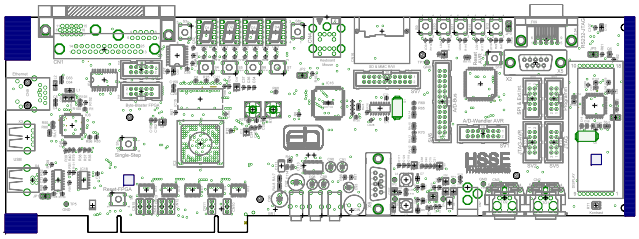


Figure 2: Silkscreen of the rapid prototyping board SandboxX.

The audio codec can be configured and is capable of transferring audio data in several different ways. For our application we used the I^2S interface to transfer audio data and the SPI interface to configure the codec. The audio codec is used in slave mode, so that the clock for the codec has to be generated with a hardware frequency divider implemented on the FPGA. This makes it easier to keep the system synchronous. The serial I^2S audio data is converted to a 24 bit parallel signal by a hardware unit. A valid bit indicates when new data arrives. Sending audio data to the audio codec works the other way round making use of a parallel to serial converter unit.

The analog audio data is pre-amplified and routed to audio connectors, so that the SandboxX can be used standalone for creating audio effects. A disadvantage of the prototyping board with regard to the analog signal quality is the low-cost power supply by an USB port. This keeps cost down, but results in a higher noise floor of the supply creeping into the audio signal domain.

The SDRAM is connected via an address / data bus with 32 data lines. Because the SDRAM is placed beside the FPGA and we use a relatively high clock frequency for the memory (48 MHz system clock), we had to shift the clock's phase with an integrated PLL of the FPGA to meet timing requirements.

The MIDI and PS/2 interfaces are connected to the FPGA via appropriate level conversion and an opto-coupler. The MIDI interface is used to connect the MIDI foot controller described in section 3.5. A PS2-compatible mouse was used during the debugging phase to change the bandpass centerfrequency of the *WahWah* effect (see section 4.5).

The configuration data for the FPGA is automatically loaded from an SPI Flash ROM on the board by a programmer implemented in a separate CPLD.

3. SYSTEM DESIGN

The aim of our work is to implement audio effects as dedicated hardware units without losing system flexibility. The system had to have the capability to be configured and parameterized easily. Our proposed solution is the use of a softcore processor leading to a hardware/software co-design as proposed in [6] which is implemented on a single chip. We used the *NIOS II* softcore processor supplied by the FPGA vendor *Altera* to run the software part of the system. The effects are implemented as dedicated hardware units to do the signal processing while the processor core and thus the software controls signal routing.

The used prototyping platform SandboxX offers only a single SDRAM chip with direct interconnect to the FPGA. This kind of memory needs a special controller in order to be accessed in a correct way. In our system the SDRAM is connected to the bus system of the softcore processor via the SDRAM controller provided by the Altera NIOS II development system. While the SDRAM cannot be directly accessed by the audio effect units which might use quite large amounts of memory (especially delay based effects) the use of software to control the audio data streams through the SDRAM posed no performance problems. The processor just has to forward data so that little processing power is needed.

For communication between the *NIOS II* processor and the audio effects we used general purpose IOs instead of

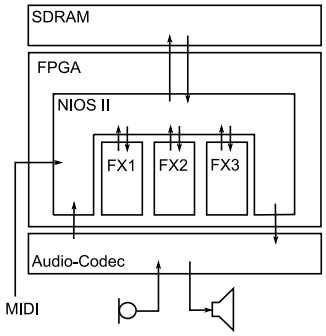


Figure 3: Hardware/Software system

integration with the processor system bus (Fig. 5). This has the advantage that the softcore processor could be easily exchanged and the interface needs less implementation effort. This approach also offers an easy access of interrupt sources. The drawback with respect to a direct system bus connection is the smaller peak performance that can be gained. In our system the throughput was by far high enough to prevent dropping of audio date in any case. In future applications the processor's system bus (*Altera Avalon Bus*) could be used together with some kind of audio stream switching matrix implemented as a dedicated hardware unit to make the system even more independent of software performance irregularities.

The software manages and controls the system to keep the trade-off between performance and flexibility. A *MIDI* control unit (e.g. foot switch) can be connected to the system. With such a device one can choose and control different effects in realtime.

The processor is the only instance that has access to the SDRAM. An effect can give the processor a request of writing data to it's effect memory or read data from it. The used hardware-software interaction scheme is depicted in Fig. 4.

The measurement of the processor usage at the memory intensive delay effect results in 33% usage for the data transport (audio codec, SDRAM) and 66% usage for the main loop.

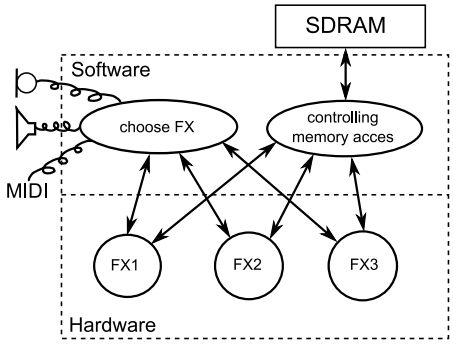


Figure 4: Hardware-software interaction scheme.

3.1. Field-Programmable Gate Arrays

Field-programmable gate arrays (FPGAs) are on the verge of revolutionizing digital signal processing in the manner that programmable digital processors (PDSPs) did nearly two decades ago [7, 8]. Many front-end digital processing algorithms, such as FFTs, FIR or IIR filters, to name just a few, previously built with ASICs or PDSPs, can now be replaced by FPGAs. Modern FPGA families provide DSP support with multipliers and fast-carry chains that are used to implement DSP algorithms at high speed, with low overhead and low costs [2].

3.2. Embedded Processor

The reason why we used the NIOS II softcore processor is the good support for custom processor system design offered by the Altera Quartus II tool set. We were able to build a specific system exactly tailored to our purpose with all the peripherals needed. They can be configured to adopt specific user needs. An example are the ports used for audio data, because they can be generated with the exact bit width delivered by the audio codec. Further, we can add as many SPI and UART interface units as we need for our peripherals (MIDI, programmable clock generator, audio codec). Detailed information concerning functionality of NIOS II peripherals is contained in [9].

The processor core was generated with the "standard" settings (NIOS II/s), featured with instruction cache, branch prediction, a hardware multiplier as well as a hardware divider. The debugging interface was generated with configuration "level 2". This means that two hardware breakpoints and data triggers are supported and debugging of the system via JTAG-interface is possible. For the hardware MIDI interface an UART module from the Altera SOPC Builder development tool was integrated into the processor system.

The system clock of the processor and all peripherals is set to 48 MHz and the internal RAM size implemented in FPGA internal RAM blocks is set to 15 KByte.

3.3. Software Development

The software for the NIOS II is written in C. It's main tasks are the parameterization of the hardware effects in reaction to the incoming MIDI data. Besides that the software is controlling the audio data flow from one effect to an other effectively acting as an audio routing matrix and also configures and initializes the audio codec. Altera offers a software development environment based on Eclipse for the NIOS II family. This environment includes a (fee free) GCC compiler optimized for the NIOS II instruction set. The SOPC Builder tool also generates processor specified libraries. Such libraries can be included into the development environment and provide the function of a hardware

abstraction layer.

Although possible, an operating system is currently not used.

3.4. Interface of Effects Units

The interface between the control unit and a single hardware effect unit plays an important role in the overall system design. The user should be provided with maximum flexibility when using the audio effects, which of course includes the way in which audio signal is routed through the effects. There are several implementation strategies, which fall into consideration:

- **Static effects-line:** This attempt leans against the technique used with the common guitar gadget boxes. It means, that there is a fixed order how the effects are joined together and the user just can switch on or off.
- **Multiplexing structure:** An expanded structure of the static effects-line could be a huge multiplexing matrix, so that the effects order is more flexible.
- **PIO:** In order to gain flexibility some software may be required. This branch stands for the easy way of connecting hardware effects to the softcore processor, namely through simple PIOs (Parallel Input/Output).
- **Avalon Bus:** This method is the advanced strategy of PIOs. We used the NIOS II softcore, which main bus system is an *Avalon Bus*. Each hardware effect may provide an Avalon Bus-connection, which is faster than PIOs.

In a software DSP system the routing of the data stream is done in a similar way the data processing is done. Because dedicated hardware units are used for audio processing the routing has to be done by a special unit that acts as a central switch node connecting all processing nodes.

Here the flexibility of the embedded NIOS II softcore comes into play. SOPC-Builder allows any number of port peripheral units needed. Consequentially each individual effect got it's own data and control ports which are directly connected to the NIOS II giving software full control of the routing of audio data. Audio inputs as well as outputs of all processing units are also connected directly to the softcore so that the user is able to pass the audio data through different effects in different order (see Fig. 3).

Some effects may require buffer memory also. Unfortunately it's neither affordable to spend each effect it's own SD-RAM nor was it possible to do so on our prototyping board. The single external SD-RAM (16 MB) is managed by the NIOS II via a SD-RAM controller peripheral. If an effect needs to buffer data, it can use the RAM-ports as shown in Fig. 5. The RAM address is a relative one, the softcore converts it and forwards the data physically to the SD-RAM.

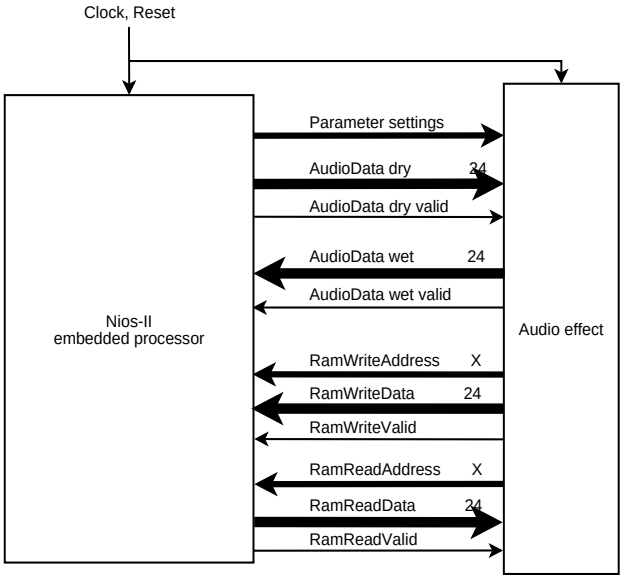


Figure 5: Interface between NIOS-II softcore and effects.

3.5. MIDI Control Interface

The human control interface is a MIDI foot controller. There are two different types of pedals on the foot controller: ten foot switches, which are used to switch through effects and two expression pedals, which are used to parameterize the active effect. The software on the NIOS II is sensitive on the used MIDI control sequences. If a valid command is received, the parameters for the chosen effect are calculated and transmitted from the processor to the hardware effects block. Changing effects disposes a recalculation of the parameters.

4. EFFECTS

4.1. Chorus

The Chorus effect simulates playing various instruments simultaneously. When more musicians play instruments simultaneously, they will not play exactly synchronously. A fixed and a variable time difference between the instruments exists. The chorus effect does the same thing. It adds an audio signal several times to itself where each instance of the audio signal (i.e. the summands) is delayed by some amount in time. The delayed signal is generated by adding a fixed delay in the range of 15 – 20 milliseconds and a sweeping delay of 4 – 8 milliseconds (see Fig. 6). In our implementation, the sweeping delay has the waveform of a triangle and

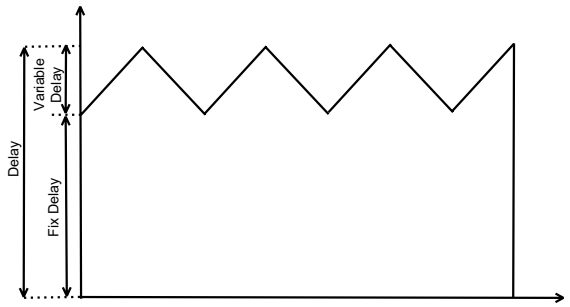


Figure 6: Delay of Chorus

a frequency of 1 – 5 Hz. Another common used waveform for the sweeping delay in the chorus effect is a sine wave. The advantage of a sine wave is that it is a very smooth function and creates a harmonic sound. The sweeping frequency is a parameter which can be changed while the system is in use. As an option there could be also used a logarithmic or sine waveform for the low frequency oscillator to change the variable delay.

4.2. Delay

Depending on delay time the effect has different names because of the particular character of the resulting sound. If the delay is in the range between 10 and 25 ms, we will hear a quick repetition named slapback or doubling. If the delay is greater than 50 ms we will hear an echo [10]. One can use a single hardware unit for both of these delay based effects while the splitting into two different effects can be done in software.

Address calculation for the RAM used as audio buffer is implemented in hardware in the manner of a ringbuffer. Via the parameter *delaytime* the size of this ringbuffer can be controlled. The output is calculated in a feedback loop. This means the output is computed by the input sample and a former output value read from the buffer. The output values are scaled with the parameter *level* and written to the buffer memory.

4.3. Echo Cancellation

An emerging topic in audio signal processing is echo cancellation. Especially the extensive use of Voice over IP, hands-free kits and conference calls enhanced the publicity and the technology of canceling interfering reflections. This problem occurs due to the fact that the spherical radiation of loudspeakers is reflected by objects. Since acoustic waves are expanding at an almost fixed speed, different time shifted reflections, depending on the place where they are measured, are created. Consequently, rooms have different acoustic characteristics, which means that e.g. a small room with reflecting walls is totally different to a big room with walls that are reflecting hardly anything.

The major problem in conjunction with these reflections arises due to the use of acoustic transmission. Because data is sent in packages there has to be a buffering at first, additionally time delays occur during the transmission. Such a delay could be a packet loss or a busy resource. Fig. 7 illustrates the problem. At first participant A is sending data, so there is a delay caused by buffering at point A as well as the transmission delay to point B. Accordingly the data is played at point B, reflected, recorded, buffered again and sent back to point A. Consequently the sender gets the interfering reflections with the sum of all delays, which is just annoying in the slightest case but can also render the phone useless.

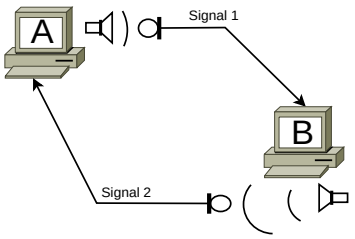


Figure 7: Delayed audio transmission.

A solution of this problem can be filtering of the reflections which are created on the side of each participant. The basic solution of this problem was already solved 50 years ago by *Norbert Wiener*. He found a way to calculate the coefficients of a filter for filtering just a specific signal. Therefore the inverse autocorrelation of the received signal, as well as the cross correlation between the received signal and the sent signal is needed:

$$h_{opt} = R_{xx}^{-1} r_{xs}^{(\lambda)} \tag{1}$$

Now it would basically be possible to measure the autocorrelation and the cross correlation in advance, for calculating the optimal impulse response to create an echo canceler. But since the reflections are different for every place and even differ when moving an object, the calculations have to be done during filtering. This requires the use of an adaptive filter.

Adaptive filters (see Fig. 8) are calculating the impulse response iteratively by minimizing the error signal. They start with an assumption for the impulse response. With this assumption the first convolution is calculated, then the result is used for generating an error signal that can be used to update the filters coefficients again. Thus, the impulse response is converging to the optimum Wiener solution with each iteration. An algorithm which is commonly used for this purpose is the NLMS, the normalized least mean square algorithm. Therefore the variable μ is used for the step-size

parameter to control the attended adaptation time as well as the residual error. The normalization of this equation is done by the use of the squared Euclidean norm. Haykin [11] describes the effect as lowering the adaption for large signals and increasing the adaptation for small signals. This produces a lower noise which stabilizes the calculation and increases the adaptation time. Since for our simulations and implementations we are using a range of values between -1 and +1, the division has to be exchanged with a multiplication to achieve the same effect. Consequently it can be

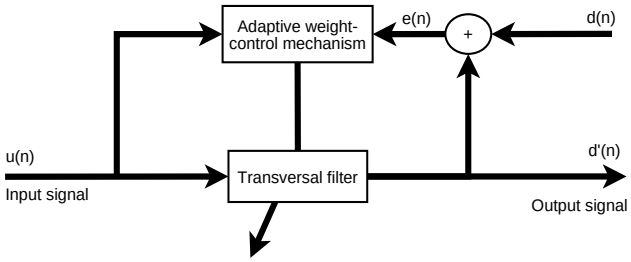


Figure 8: Adaptive filter

summarized that the calculation effort takes about M multiplications and M additions for the convolution of the FIR filter and additionally $3M$ multiplications and M additions for the recursive algorithm.

$$\widehat{w}(n+1) = \widehat{w}(n) + \frac{\tilde{\mu}}{||u(n)||^2} u(n) e^*(n) \qquad (2)$$

To achieve a good performance it should be considered that the length of the filter has to be at least as long as the reflection takes to be recorded. Assuming a delay path of 10 meters it takes about 30 ms until the delay will be recognized. This leads to a minimal order of

$$M = 30\,\mathrm{ms} \times 44117 \frac{\mathrm{Values}}{\mathrm{s}} = 1300$$

for using audio quality sample rates. Such a high order goes along with very high demands for the processing speed, since it takes about $6M$=7800 calculations per value, or $340M$ calculations per second, for using CD quality sample rate. Based on the fact that the used IC is working with a clock of 48 MHz it can be assumed that

$$steps = \frac{48MHz}{44117Hz} = 1088$$

steps are needed to calculate the adaptive filter between two arriving audio values. This already shows that the implementation could be a challenge, because the proportion between calculations and time is 7:1. So in order to solve this problem the parallel advantages of the FPGA have to be used. This can be done by dividing the filter into several parts, calculating them separately at the same time and combining them finally together. This possible realization differs from a digital signal processor in that way, that the maximum number of fragmentation is much higher and that the unused hardware parts are not affected at all. The used IC offers for example 35 embedded 18 bit multiplier. Furthermore a VHDL simulation shows that in order to realize 350M calculations in a second, the filter has to be divided in 5 different parts which would just just 15% of the available multiplier. And while the resources of a DSP would have been depleted by this filter, the implementation on a FPGA would still have a lot of calculation power left.

4.4. Flanger

The flanger is a delay based effect like the echo, so the architecture of the delay effect can be used as a starting point. The difference is that in the echo effect a linear addressing scheme is used while the flanger effect uses a delay time that varies at a low frequency. The various frequencies differ with steps by $\frac{1}{44{,}1kHz}$, thus no fractional interpolation is needed. We create a sinusoidal oscillation via a direct digital synthesis (*DDS*) unit. DDS is a technique using a look-up table to store the values of a function which should be generated. This oscillation is added to the linear addressing used for the ringbuffer as shown in Fig. 9. The frequency can be controlled by a parameter.

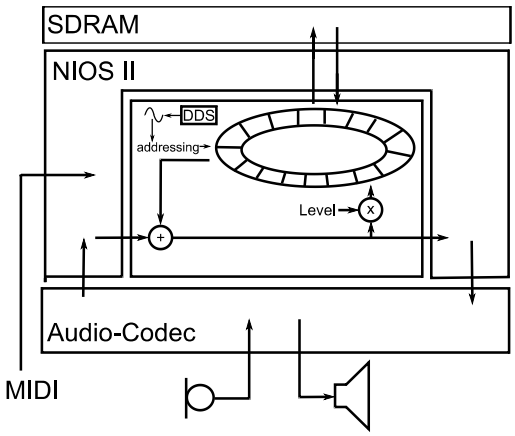


Figure 9: Flanger effect architecture.

4.5. WahWah

The WahWah effect describes a time-varying bandpass filter. In this case the concept "second-order allpass" as found in [10, p. 41] was realized. The idea is to create an allpass filter with phase shifting by 360. The filtered signal has to be *subtracted* from the original signal. When this elementary operation is done an allpass behavior results instead of a bandpass (BP) behavior. Analogue to this fact, we can *add*

the original signal to the filtered signal so that the result is a band reject (BR) filter as seen in Fig. 10.

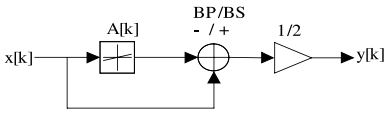


Figure 10: Blockdiagram of the allpass method

4.5.1. System Behavior

The cut-off frequency is the frequency at which the phase is shifted by 180. When we consider the complete system including subtraction, exactly the same point represents the currently center frequency of the bandpass. The frequency band for the complete phaseshift (-360) is the bandwidth of the bandpass (see Fig. 11).

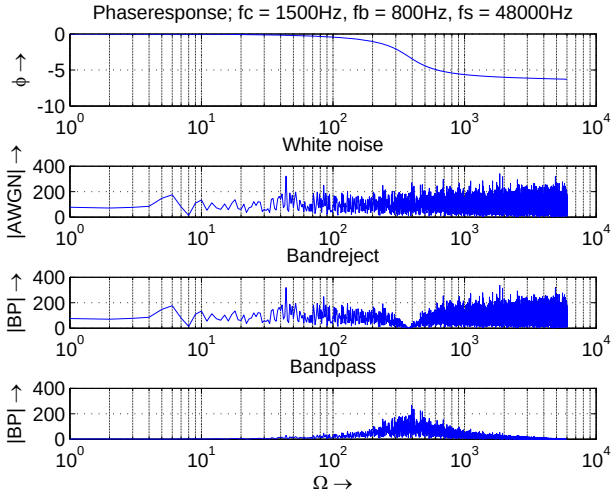


Figure 11: Demonstration of the allpass-method

Because the used filter is an IIR type of second order, there are just 6 filter-coefficients. In addition there is the special character of an allpass, which means that the infinite filtercoefficients are mirrored to the finite filtercoefficients (equation 5). Also the first infinite filtercoefficient has always the value 1. Accordingly we just need to calculate 2 coefficients. Parameter c describes the bandwidth of the bandpass, whereas parameter d describes the center frequency.

$$c = \frac{tan(\pi f_b/f_s) - 1}{tan(\pi f_b/f_s) + 1} \tag{3}$$

$$d = -cos\left(2\pi \frac{f_c}{f_s}\right) \tag{4}$$

$$h_{Allpass}(z) = \frac{-c + d(1-c)\cdot z^{-1} + z^{-2}}{1 + d(1-c)\cdot z^{-1} - c\cdot z^{-2}} \tag{5}$$

One last problem which needs to be handled is the time-variance. The user of the effects-processor should be able to change the center frequency of the WahWah bandpass in realtime. The filter coefficients therefore need to be calculated depending on the desired center frequency. This operation would require the calculation of the cosine (see equation 4). A compromise was to pre-calculate a few possible coefficients and store them in a lookup table. Because these lookup tables don't need to be changed during system uptime, we used the FPGA's Block-RAM as ROM for this matter.

4.5.2. Hardware Effort

The WahWah effect requires the implementation of a second-order IIR filter. Therefore each order is represented by a VHDL process including forward and recursive branch. 4 filter coefficients out of 6 are not equal to one, therefore the filter needs 4 multiplications per sample. The Cyclone II FPGA includes 35 multipliers with a bitwidth of 18 bits. In order to gain maximum speed the multiplications are done in parallel which means, that 4 multipliers are used.

As said above the pre-calculated filter coefficients need to be saved. For this matter the FPGA's internal memory blocks are used. Considering the fact, that allpass-filter coefficients are mirrored (see [10]) and that the bandwidth is a constant there is just one filter coefficient left, which need to be saved. This single coefficient represents the center frequency and was pre-calculated for a frequency range of 200Hz to 2kHz considering, that in the end 1024 24-Bit values are ready to use. This results in 18432 bits of required memory space. The hardware unit itself uses about 400 logic elements containing 250 registers. The maximum achievable clock frequency is 60 MHz.

5. ADVANTAGES, DISADVANTAGES TO A DIGITAL SIGNAL PROCESSOR

The main advantage in implementing audio effects with as dedicated hardware is the very high throughput and low latency. Dedicated hardware also avoids the sometimes unpredictable behavior of a software based implementation. Because all the hardware blocks are working in parallel, we can implement complex designs and as long as we have enough free space on the chip, computing power is guaranteed. This offers us the possibility to have a huge range of effects working in parallel, because each effect works independently from the others. If we want to implement our

effects "the traditional way" by using a digital signal processor, we sooner or later will reach the calculation power limit.

Some applications can be implemented easier in software. In our case it was the control of the effects with the MIDI pedal. Therefore we decided to use also a softcore processor for managing the effects. Managing in this point means to parameterize them. Implementing also the audio effects in software (e.g. on a DSP) would have been easier than in hardware. There are currently more example-implementations available and the design process is faster in software. Also testing would have been easier, because it takes a lot of time to build a test environment in VHDL for hardware effects.

One large disadvantage of FPGAs is left to be mentioned. The point is the multiplication. Common FPGAs seldom provide DSP elements with more than 20 bits width. But digital signal processors mainly calculate with an accuracy of 32 bits. Some DSPs (e.g. *SHARC* from *Analog Devices*) even provides a 40 bit accuracy when multiplication is done. For high-end audio the bitwidth is crucial and must not be neglected. When we want to use the same broad bitwidth on FPGAs it possible to cascade the DSP elements which will in turn increase the propagation delays through combinatorics and makes the system working slower.

Additionally there are a number of well established libraries for DSP processors available, which ease and fasten the development. But on the contrary there are not that many IPs available which serve our specific needs. Furthermore, the advantage of these libraries is that they are available for free and thus have been used, tested and improved by a huge community.

6. CONCLUSIONS

The hardware/software co-design method described in this paper provides a practical alternative to the software centric systems dominating the market today. While dedicated hardware units used for audio realtime processing offer optimum performance in terms of latency and throughput the use of an associated software unit can still take care of the parameterization of the system and provides the flexibility a user is accustomed to.

The hardware/software co-design approach we have chosen has proved to be a practical way for the realization of even complex audio realtime effects units. Still there's lots of work left to be done. Varying bit widths throughout the course of the audio processing assembly line would be easily implementable using dedicated hardware. Among the topics we would like to address in future work are the implementation of digital audio interface standards (e.g. MADI) with special emphasis on an efficient solution for the all-pervasive problem of synchronization and the implementation of audio compression algorithms such as FLAC in dedicated hardware.

7. REFERENCES

[1] Klaus ten Hagen, *Abstrakte Modellierung digitaler Schaltungen*, Springer-Verlag, Berlin, Heidelberg, NewYork, 1995.

[2] Uwe Meyer-Bäse, *Digital Signal Processing with Field Programmable Gate Arrays*, Signals and Communication Technology. Springer-Verlag, Berlin, Heidelberg, NewYork, 2 edition, 2007, ISBN-13 978-3540211198.

[3] Wolfgang Pauli, Markus Pfaff, and Stefan Reichör, "Dsp in dedicated hardware: Raising value abstraction for fixed point implementation," in *International Symposium on Signals, Systems, and Electronics ISSSE 04*, Linz, Austria, August 2004, University of Linz, ISBN 3-9501491-3-9.

[4] Mario Huemer, Michael Lunglmayr, and Markus Pfaff, "A lecture course series: From concept engineering to implementation of signal processing algorithms with FPGAS," in *Proceedings of the 13th European Signal Processing Conference EUSIPCO 2005*, Antalya, September 2005, Istanbul Technical University.

[5] D. Malzner, J. Seifert, J. Traxler, H. Weber, and G. Wiendl, "Dafx project homepage," http://www.dafx-hsse.info, 2006.

[6] G. Truhlar, T. Pühringer, G. Schedelberger, M. Pfaff, and J. Langer, "Hardware/software co-design of a realtime-rendering architecture for embedded systems," in *Austrochip 2004*, Villach, Austria, October 2004, Technikum Kärnten.

[7] Brian Dipert, "FPGAs DiSPlay their processing prowess," *EDN Magazine*, , no. 22, pp. 61–68, October 2002, Avaiable online from http://www.edn.com.

[8] Nick Tredennick, "The death of DSP," http://www.ttivanguard.com/dublin/dspdealth.pdf, August 2000.

[9] Altera, *NIOS II Processor Handbook*.

[10] Udo Zölzer, Ed., *Digital Audio Effects*, John Wiley & Sons, Inc., New York, 1 edition, 2002, ISBN-13 978-0471490784.

[11] Simon Haykin, *Adaptive Filter Theory*, Prentice Hall, New Jersey, 2002, 0-13-090126-1.

BINAURAL SOURCE SEPARATION IN NON-IDEAL REVERBERANT ENVIRONMENTS

Sylvia Schulz and Thorsten Herfet

Telecommunications Lab
Saarland University, Germany
{schulz, herfet}@nt.uni-saarland.de

ABSTRACT

This paper proposes a framework for separating several speech sources in non-ideal, reverberant environments. A movable human dummy head residing in a normal office room is used to model the conditions humans experience when listening to complex auditory scenes. Before the source separation takes place the human dummy head explores the auditory scene and extracts characteristics the same way as humans would do, when entering a new auditory scene. These extracted features are used to support several source separation algorithms that are carried out in parallel. Each of these algorithms estimates a binary time-frequency mask to separate the sources. A combination stage infers a final estimate of the binary mask to demix the source of interest. The presented results show good separation capabilities in auditory scenes consisting of several speech sources.

1. INTRODUCTION

Humans are masters in analyzing their auditory environment and in separating different sound sources. Consider the classical cocktail party example, where several people are talking simultaneously in the same room. Humans have no difficulty to attend to a single person while ignoring all the other people, additional artificial sources and background noise. Today's computational approaches for source separation – especially in reverberant environments – are far from achieving this extraordinary ability of the human brain.

When humans enter an auditory scene they first look around and estimate several features of the environment around. When source separation is required – i.e. when starting a conversation with another person – this knowledge is used to enhance the separation process. The presented source separation framework tries to model this human behavior to enhance the following source separation. To imitate the human listening situation, a robotic human dummy head, called Bob, is used. Bob resides in a normal office room of size 10×6 m and a reverberation time $RT_{60} = 0.4\ s$ and is able to move in three degrees of freedom to explore the auditory scene around him. A conventional 7.1. loudspeaker installation is utilized to construct an auditory scene consisting of several spatially separated sources by assigning each source to a specific loudspeaker. The auditory scene around is recorded via microphones in Bob's ears.

2. TIME-FREQUENCY MASKS

Rickard et al. [1] showed that speech signals are sparsely distributed in high-resolution time-frequency (TF) representations. TF representations of different speech signals overlap only in few points and so are approximately orthogonal to each other. This approximate orthogonality in the TF-domain justifies the use of TF-masks that emphasize regions of the TF-spectrum that are dominated by a specific source and attenuate regions dominated by other sources or noise. Masking effects in the human auditory system motivate the use of binary TF-masks: Within a critical bandwidth humans don't recognize sounds that are masked by louder sounds.

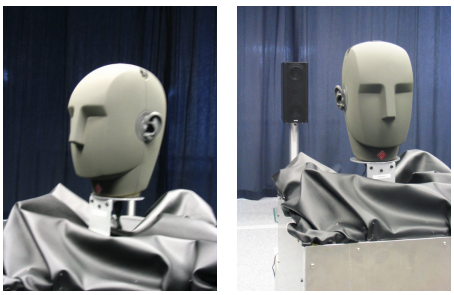


Figure 1: *Bob – The robotic head.*

Several researchers in computational source separation suggest an ideal binary mask as final goal of computational source separation algorithms (i.e.[2], [1]). Brungart et al. [3] support this goal by noting that the intelligibility of separated sounds increases if more and more energy of the ideal binary mask is reconstructed.

Assume $s_i(t,f)$ denotes the energy of the target signal$_i$ in TF-bin at time t and frequency f and $n_j(t,f)$ denotes the energy of the j-th interfering signal in this TF-bin. The ideal binary mask $\Omega_i(t,f)$ for target source$_i$ and a threshold of 0 dB is defined as follows:

$$\Omega_i(t,f) = \begin{cases} 1 & s_i(t,f) - n_j(t,f) > 0 \quad \forall j \\ 0 & \text{else} \end{cases} \tag{1}$$

2.1. Short-Time-Fourier-Transform

A commonly used TF-representation is the lossless and computationally efficient Short-Time-Fourier-Transform (STFT). The discrete STFT analyzes the time-domain signal in linearly spaced frequency channels up to the Shannon frequency. For a general discrete signal $x(n)$ and an arbitrary discrete analysis window function $w(n)$ the STFT is defined $\forall q \in \{0,1,...,N-1\}$ as

$$X(k,q) = \frac{1}{\sqrt{N}} \cdot \sum_{n=0}^{N-1} w(n)x(n+k)e^{-i2\pi\frac{qn}{N}}. \tag{2}$$

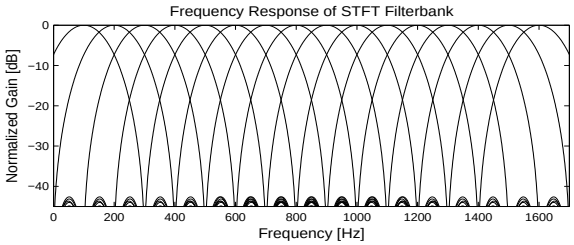


Figure 2: *Frequency Response of STFT using a Hamming window.*

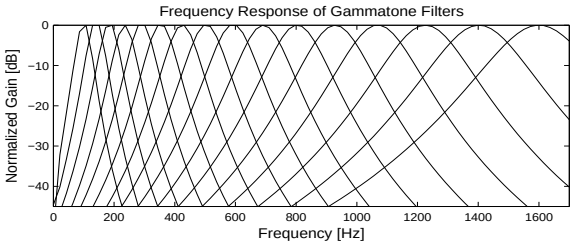


Figure 3: *Frequency Response of Gammatone Filterbank.*

Figure 2 shows the positive frequency response of the STFT for a Hamming window of length 32 using a sampling frequency of 3.2 kHz. The shape of the linearly spaced filter channels and the overlap between two consecutive channels is specified by the shape of the analyis window function.

Yilmaz et al. [1] showed that the approximate orthogonality of different speech sources in the discrete STFT representation with Hamming windows of 64 ms length is satisfied and the STFT spectrum is a suitable and easy representation for assigning complete time-frequency regions to specific sources.

Besides an amplitude and phase estimate for each bin in the spectrum, the STFT provides no further low-level information about this bin that could be used to infer the dominating source. Because of the limited time and frequency resolution the estimates are only coarse and averaged over the complete analysis window. If a specific bin is dominated by one source, there may also be energy of other sources in this bin which severely forge the amplitude and phase estimates.

Almost all energy of speech signals is distributed in frequencies up to 8 kHz. For analysing speech signals, a finer frequency resolution in the low frequency range is favorable, whereas in higher frequencies a coarse resolution is sufficient. Because the STFT analyses linearly up to the Shannon frequency, the frequency resolution in the low frequencies cannot be enhanced by increasing the sampling rate.

A source separation algorithm should use all information about a specific time-frequency region to increase the possibility of correct assignment. Using only an amplitude and a phase estimate for the assignment decision of a complete STFT-bin is quite limited and is not very reliable in reverberant mixtures. To enhance the decision process more information about each STFT-bin must be examined.

2.2. Cochleagram

Many source separation architectures try to imitate the frequency analysis of the human auditory system. The frequency analysis of the human cochlea can be approximated using a bank of gammatone filters. The impulse response of a gammatone filter is defined as the product of a gamma function and a tone [4]:

$$g_{f_c}(t) = t^{N-1} e^{-2\pi b(f_c)} \cdot \cos(2\pi f_c t + \phi) \quad \forall t \geq 0 \qquad (3)$$

where N denotes the order of the filter and f_c denotes the center frequency of the filter. The value $b(f)$ determines the bandwidth of the filter and is usually set to the equivalent rectangular bandwidth (ERB) of human auditory filters. A bank of such gammatone filters gives a good fit to experimentally derived estimates of the frequency analysis of the human cochlea and for such the TF-representation of a gammatone filter bank is commonly called cochleagram.

Figure 3 illustrates the frequency response of a bank of 16 gammatone filters in the frequency range from 100 - 1600 Hz. Consecutive filters are spaced logarithmically on the frequency scale. Filter channels in the low frequencies have fine frequency resolution, but coarse time resolution. Conversely the high frequency channels have coarse frequency resolution, but fine time resolution. The coarse time-resolution in the low frequencies is acceptable as signals consisting of low frequencies change slowly, whereas high-frequency signals need finer time-resolution to illustrate the rapid changes.

The inversion of a given cochleagram to a time-domain signal is non-trivial and lossy. There exist some approaches that yield quite good invertion results (i.e. [4], [5]), but these are complex to compute and only approximately orthogonal, which results in non-perfect reconstruction.

3. OVERALL ARCHITECTURE

The STFT is easy and lossless to compute, but the filter channels are positioned linear on the frequency scale which yields only a coarse frequency resolution in the important low frequencies. Also the amplitude and phase information are averaged over the complete analysis window and so not really reliable in reverberant environments.

The cochleagram on the other hand analyses the signal with logarithmically spaced filter channels and allows a finer frequency resolution in the low frequencies, but the invertion of a given cochleagram to a time-domain signal is quite complex.

The source separation framework presented in this paper combines the positive features of the STFT with the positive features of the cochleagram while eliminating some of the negative features. The overall goal of the source separation is to find the ideal STFT-mask. The core source separation process however is based on the analysis of the corresponding region in an additionally computed cochleagram. This way the macroscopic STFT-transform is used to define the demixing masks and to finally demix the original sources. The core assignment of each STFT-bin to a specific source is based on the corresponding region in the microscopic cochleagram and is only supported by the information gained from the STFT-spectrum.

This proceeding is analog to the approaches used in MPEG audio coders. For example MPEG Audio Layer 3 uses a FFT of 1024 samples to analyse the input signal and to apply the psychoacoustic models. The critical subsampling however is realized using only 32 subbands [6].

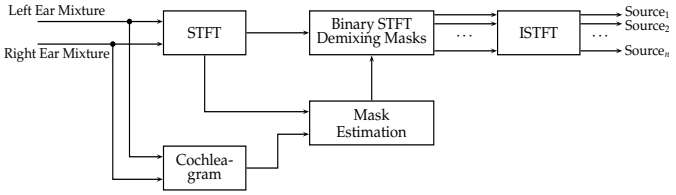


Figure 4: *Overall architecture for source separation framework.*

Figure 4 illustrates the system architecture of the source separation framework. The incoming signals of the left and right ear are STF-transformed and the respective cochleagram of each ear signal is computed. The mask estimation process computes for each source a binary STFT-mask based on the information gathered from the detailed cochleagram and supported by coarse information of the STFT spectrum. Finally the STFT spectrum is multiplied by each binary STFT mask and is transformed back to the time-domain, yielding the demixed time-domain signals.

The mask estimation stage tries to make use of all information that could be established using standard or sophisticated signal processing methods. In a first step Bob, the movable human dummy head, analyses the auditory scene and identifies the position of the preferred speaker in the room. In further steps this information is used to enhance the source separation, that uses both interaural and monaural cues to distinguish the TF-bins. Because of reverberation many of the cues used to separate bins are distorted and can only be used to some extent. To face the reflections and reverberations several algorithms compute independent estimates of the binary masks. In a final stage the estimated masks of each algorithm are combined to find a best estimate.

4. AUDITORY SCENE EXPLORATION

When humans enter an auditory scene such as a cocktail party, they automatically analyse the environment around them. Humans recognize the number of possible sound emanating sources, classify them according to speech or artificial sounds and estimate or recall from memory several expected features of each source. When a communication between the human and one of the sources begins, much information is already known to human cognition and is used to support and enhance the separation process.

The source separation architecture presented in this paper tries to mimic these cognitive abilities of the human brain. So prior to separating the speech sources, Bob analyses the auditory scene and estimates several parameters that can be used to enhance or enable later separation approaches. The following separation algorithms expect as input the position of the source to be enhanced in the azimuth plane. Furthermore some of the separation schemes require an estimate of the fundamental frequency of the desired speech source.

4.1. Source Localization

In the following sections and the source separation algorithms presented later the source of interest is assumed to be the – in some sense – strongest source in the auditory scene. The localization of the desired source is realized using an adaptive estimate of the interaural time differences between the two ears.

The interaural time difference (ITD) – the arrival time difference between the left and right ear signal – is used as localization cue and is estimated based on the correlation between the two signals. Assume x_L and x_R denote the time domain signal of the left and the right ear. The correlation function is defined as

$$R_{x_L x_R}(l) = \sum_{t=t_s}^{t_e} x_L(t+l) \cdot x_R(t). \tag{4}$$

Each source in the auditory scene contributes a peak in the correlation function. Further peaks can be introduced by reflections and reverberations. Detecting the highest peak in $R_{x_L x_R}$ yields a first estimate of the incidence direction of the strongest source, so the movable human dummy head Bob turns to this estimated position.

Because of the reverberation and interference Bob cannot rely on the validity of the estimated position. Therefore a further correlation at the new position is computed that should in the ideal case have its highest peak at the position of zero degree. To account to the reverberant environment the position is regarded to be confirmed if one of the highest peaks is located near zero degree. If this peak deviates from zero with only some degrees, Bob enhances the located position. This procedure is iterated until a stable position is reached and the regarded peak of the correlation function appears at approximately zero degree. Then Bob directly faces the source of interest which is now centered around 0° relative to Bob's facing direction.

Because the specific resonances of the human ear and head are not used yet, Bob cannot distinguish between front and back only from analysing the ear signals. Possible front-back confusions are resolved by slightly moving the head to one side at the final position and measuring the direction of change of the ITD between the two ears. If it turns out that Bob has mistaken the direction, Bob turns 180° around and faces the correct source.

For a detailed description of the design, implementation and results of the source localization scheme consider the work of Haschke [7].

4.2. Fundamental Frequency Estimation

Humans tend to emit frequencies that are an integer multiple of their own fundamental frequency (F0). Especially voiced parts of speech contain most of the energy in the harmonics of F0. Source separation approaches can use the F0 to determine those frequencies that are mainly used by a speaker.

The used fundamental frequency estimation relies on an algorithm known as "Robust Algorithm for Pitch Tracking" (RAPT) [8] and determines the F0 of spoken utterances as a function of time. The time-domain signal is split in time-frames of length 5 ms and for each frame a F0-estimate is computed based on the autocorrelation of the corresponding signal.

RAPT is originally designed to work in anechoic, single source recordings and computes reliably an estimate of the F0-track and the first harmonics. In reverberant multi source recordings – such as the recordings of Bob's ears – the estimation process severely degrades and the result forms a mixture of each F0-track and additional noise.

Assuming the source of interest is directly in front of Bob and the interfering sources are distributed at other positions, the preferred signal can be enhanced by applying simple beamforming: The right and the left ear signal are summed and divided by two.

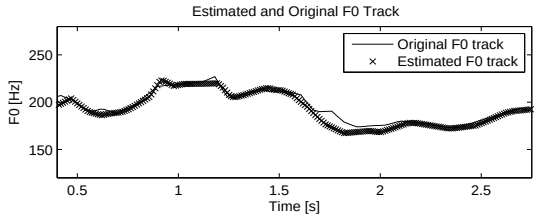


Figure 5: *Original F0-track and reconstructed F0-track for a mixture of two speech sources.*

This way the resulting signal emphasizes the preferred source and further smears the interfering sources.

The enhanced signal is used as input to the RAPT to get an estimate of the F0-track of the preferred source. For each time-frame the F0-estimate produced by RAPT is considered valid, if corresponding estimates are found in several higher harmonics. The final F0-track is constructed by linear interpolation of the valid F0s.

Figure 5 shows the result of the F0-track reconstruction for an auditory scene consisting of two speech sources at positions 0° (Bob has already geared towards the source) and an interfering source at position 45° to the right.

5. SEPARATION ALGORITHMS

Existing source separation approaches can be broadly classified into two categories:

- **Separation algorithms based on Interaural Cues** use interaural time and level differences to separate the sources. In ideal anechoic mixtures the direction of each TF-bin can be estimated and the bin is easily assigned to the correct source. Unfortunately echoic recordings blur and distort these interaural cues, so separation capabilities decrease.
- **Separation algorithms based on Monaural Cues** use only characteristics that are specific to a single signal and do not rely on the differences between the left and right ear. These algorithms mostly use the fundamental frequency of the speaker as a main feature to separate the sources.

The following algorithms assume that Bob has already analyzed the auditory scene and has turned towards the preferred source. Furthermore he has estimated the F0-track of the source of interest as described before. Imitating the human behavior, Bob automatically aligns his head to the source of interest. The goal of the following source separation algorithms is to enhance a specific source of interest, not to separate all sources.

5.1. Separation based on Interaural Time Differences

Interaural Time Differences between the left and right ear signal are used to examine the position of the respective source of each STFT-bin. Because the STFT phase value is not necessarily reliable as discussed previously, the ITD is estimated using the cochleagram. Let $X_{L_{stft}}$ and $X_{R_{stft}}$ denote the STFT-representation of the left and right ear signal and $X_{L_{co}}$ and $X_{R_{co}}$ the corresponding cochleagram representations. For each STFT-bin the corresponding left and right TF-windows $W_{L_{co}}$ and $W_{R_{co}}$ are cut out of the cochleagram. The ITD estimates of $W_{L_{co}}$ and $W_{R_{co}}$ are computed using a running cross-correlation across the time-dimension of the time-frequency regions: $\forall l \in \{-\mathrm{maxLag}, \mathrm{maxLag}\}$

$$R_{W_{L_{co}} W_{R_{co}}}(l) = \sum_{t=t_s}^{t_e} \sum_{f=f_s}^{f_e} W_{L_{co}}(t+l, f) \cdot W_{R_{co}}(t, f) \quad (5)$$

The highest peak of $R_{W_{L_{co}} W_{R_{co}}}$ yields the best estimates of the ITD for this bin. Because of reverberation and reflections there could be further peaks in the correlation function that could refer to the correct ITD and therefore should be considered. According to Faller and Merimaa [9], the height of the peak in the correlation function is a measure of reliability: The higher the peak, the more reliable the ITD estimation.

Knowing that the preferred source is at azimuth zero degree, the ITD computations offer the following three algorithms to estimate the ideal binary masks:

Algorithm 1 Assign to the source of interest all TF-bins where the estimated ITD of the highest peak of the correlation function yields an angle of incidence that deviates not more than δ° from 0° and the height of the peak is greater than h.

Algorithm 2 Assign to the source of interest all TF-bins where the estimated ITD of an existent second highest peak of the correlation function yields an angle of incidence that deviates not more than δ° from 0° and the height of the peak is greater than h.

Algorithm 3 Assign to the source of interest all TF-bins where the estimated phase of the STFT bin yields an angle of incidence that deviates not more than δ° from 0°.

Each algorithm regards only these STFT-bins that contain more energy than a specific threshold. To compare the results of each algorithm, the estimated masks are compared with an ideal mask, which is estimated from recordings of the single sources under reverberant conditions. Because reverberation differs slightly between recordings with only one source and recordings with several sources, this ideal mask is only an approximation of the real ideal mask. Using this estimated ideal mask as ground-truth there are three evaluation criteria for each algorithm:

1. *The percentage of recovered energy* of the ideal mask. The higher this percentage, the more energy of the original signal is recovered and the speech intelligibility of the desired source increases.
2. *The percentage of false estimated bins* denotes the relative number of bins that are wrongly assigned to the preferred source. According to the ideal masks, these bins should be assigned to one of the other sources of the auditory scene, as the absolute value of energy contribution to this bin of another source is larger than the energy contribution of the desired source. The lower this value, the less artifacts from other speech sources are contained in the estimated mask.
3. *The percentage of correct estimated bins* clarifies how much of the estimated bins are correctly assigned to the source of interest. The gap between this value and the percentage of false estimated bins indicates the number of those TF-bins that happen to have high energy in the recorded mixture, but none of the ideal masks of the single sources exhibit

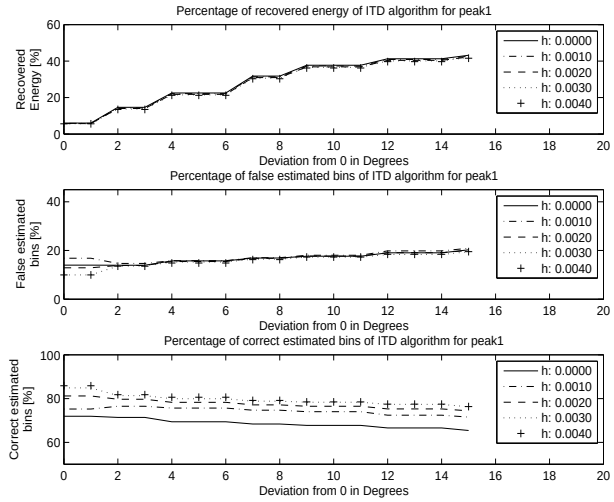


Figure 6: *Percentage of recovered energy of ideal mask, false estimated bins and correct estimated bins for the separation algorithm based on ITD and highest peak (Algorithm 1).*

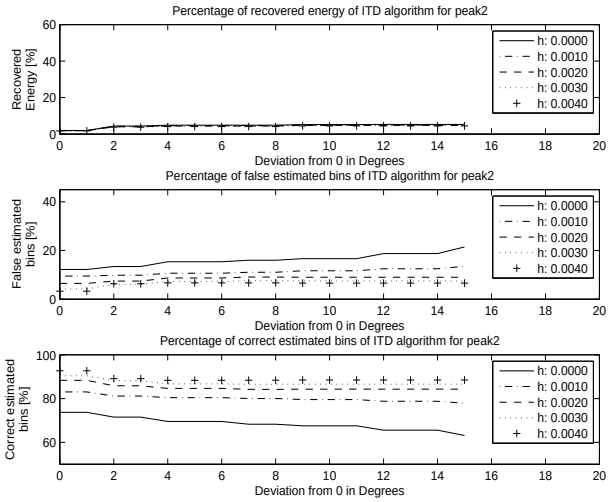


Figure 7: *Percentage of recovered energy of ideal mask, false estimated bins and correct estimated bins for the separation algorithm based on ITD and second highest peak (Algorithm 2).*

high energy in this TF-region. So these bins are likely to occur from reverberations and cannot be assigned to a specific source.

Figure 6 shows the results of algorithm 1 for different δ and h for an auditory scene consisting of two speech sources. Speech source one is located directly before the head at 0° as Bob has already geared to the source and the second source is located at 45° to the right. One can clearly see that the percentage of recovered energy increases if the deviation from zero increases. The ITDs of most of the correct TF-bins deviate considerably from the real position at 0°. In contrast to the percentage of reconstructed energy and the number of false estimated TF-bins, the percentage of correct estimated bins increases according to the peak height. TF-bins with high energy – which mostly result in high correlation peaks – are more likely to yield a correct ITD estimation as opposed to low energy bins. To achieve a good tradeoff a δ between 8 and 12 degree and high peak height h is favorable.

The same results for algorithm 2 are illustrated in figure 7. The percentage of reconstructed energy is much lower than in the case of algorithm 1. This low percentage is due to the fact that a second peak in the correlation function in most cases only exists for TF-bins at high frequencies where the correlation analysis window becomes bigger than the period of this bin. Those high frequency bins naturally include lower energy than low frequency bins, so the overall recovery is quite low.

Figure 8 plots the results for algorithm 3. The number of false estimated bins is approximately constant and the percentage of correct estimated bins decreases very slowly. As also seen in algorithm 1, the phase values of the bins deviate quite a lot from the ideal position due to reflections and interference from the other sources.

5.2. Separation based on Fundamental Frequency

If two persons speaking have a considerable different F0, their harmonics do not overlap in many frequencies. If the F0 of the preferred speaker is known in advance, this information can be used to assign the TF-bins. Each bin with a frequency value near a multiple of the fundamental frequency is more probable to belong to the preferred source, than it is to belong to one of the other sources. If additionally the F0s of the other speakers are known, the distances of the harmonics of the preferred speaker to the nearest harmonics of the other speaker can be computed and used to find the frequencies at which only the preferred speaker is present. The following F0-based algorithms are examined in the source separation architecture:

Algorithm 4 Assign to the source of interest all TF-bins where the frequency of the current STFT-bin deviates by no more than Δf Hz from the nearest harmonic of the preferred source's mean F0. If the F0 of the interfering sources is known, also the distance from the nearest interfering harmonic is used to segregate the TF-bins.

Algorithm 5 Assign to the source of interest all TF-bins where the frequency of the current STFT-bin deviates by no more than Δf Hz from the nearest harmonic of the preferred source's mean F0 and the energy in the cochleagram at the corresponding TF-unit is larger than a threshold E.

Algorithm 6 Assign to the source of interest all TF-bins where the frequency of the current STFT-bin deviates by no more than Δf Hz from the nearest harmonic of the preferred source's current F0 estimate. If the F0 of the interfering sources is known, also the distance from the nearest interfering harmonic is used to segregate the TF-bins.

Algorithm 7 Assign to the source of interest all TF-bins where the frequency of the current STFT-bin deviates by no more than Δf Hz from the nearest harmonic of the preferred source's current F0 estimate and the energy in the cochleagram at the corresponding TF-unit is larger than a threshold E.

Figure 9 displays the results of algorithm 4. The percentage of recovered energy grows as the maximal distance of the nearest harmonic of the preferred speaker grows as more and more bins

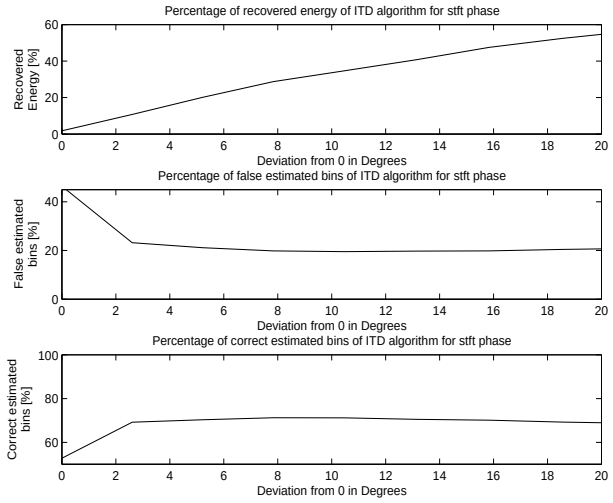


Figure 8: *Percentage of recovered energy of ideal mask, false estimated bins and correct estimated bins for the separation algorithm based on the STFT phase (Algorithm 3).*

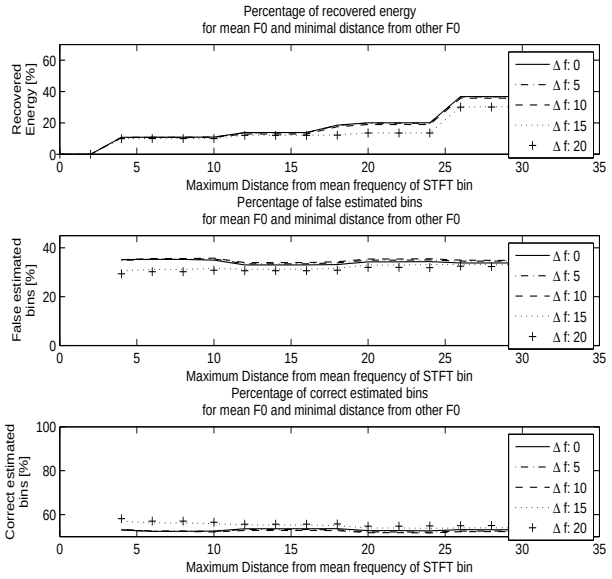


Figure 9: *Performance of the separation algorithm based on known mean F0 and the distance from the harmonic to the mean frequency of the current STFT bin (Algorithm 4). Shown are several curves for different frequency distances of the nearest interfering harmonic of the interfering speaker.*

are considered. Knowing the mean F0 of the interfering source has only minor effects that could be neglected. The rate of the correct estimated bins is constantly very low. The bad results of this algorithm arise from the fact that during a spoken word, the F0 of a human is not constant and varies about several Hz. So if the source separation relies only on the mean F0, higher order harmonics are computed incorrectly and the source separation capabilities decrease. If the complete track of the fundamental frequency is known, the separation algorithms discussed above can be enhanced.

If the absolute energy value of the regarded harmonic in the corresponding cochleagram window is used, the number of false bins and the percentage of correct bins can be slightly enhanced. Figure 10 illustrates the evaluation of algorithm 5. The higher the energy in the corresponding frequency, the higher the probability that the considered bin belongs to the preferred source. These results contribute to the reverberant environment: TF-bins with high energy and corresponding F0-characteristics are likely to originate from the main incidence direction and not from a disturbing reflection.

Figure 11 and 12 show the same results for algorithms 6 and 7, but using a complete F0 track instead of a mean value. The rate of the false estimated bins is about 10% lower than in the mean F0 case. Also the percentage of correct estimated bins is higher compared to using only an average F0. Assuming that the directly incident TF-bins have high energy, a minimum energy threshold can enhance the percentage of correct estimated bins by up to 30%. Optimal values can be achieved by using quite large maximum distances of 20 to 30 Hz.

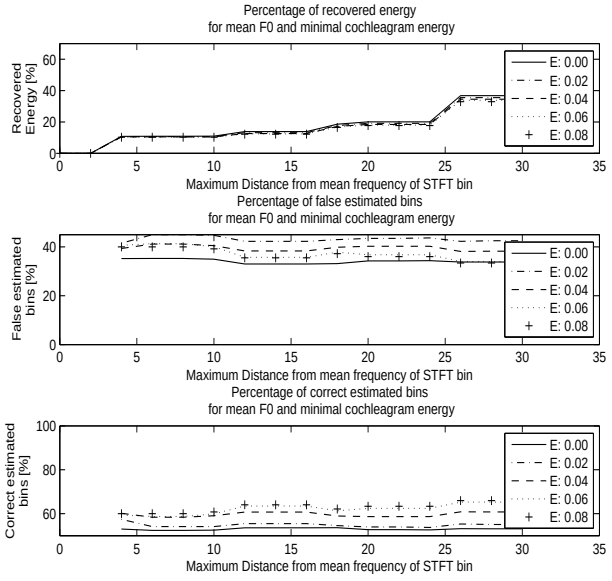


Figure 10: *Performance of separation algorithm based on known mean F0 and the distance from the harmonic to the mean frequency of the current STFT bin (Algorithm 5). Shown are several curves for different cochleagram energy levels E.*

6. COMBINING OF ALGORITHMS

Each of the introduced algorithms yields a reconstructed preferred speech source with a low to intermediate intelligibility. To enhance the separation capabilities, the algorithms work together to combine their information regarding each TF-bin. In a first stage each discussed algorithm separately estimates a STFT-demixing

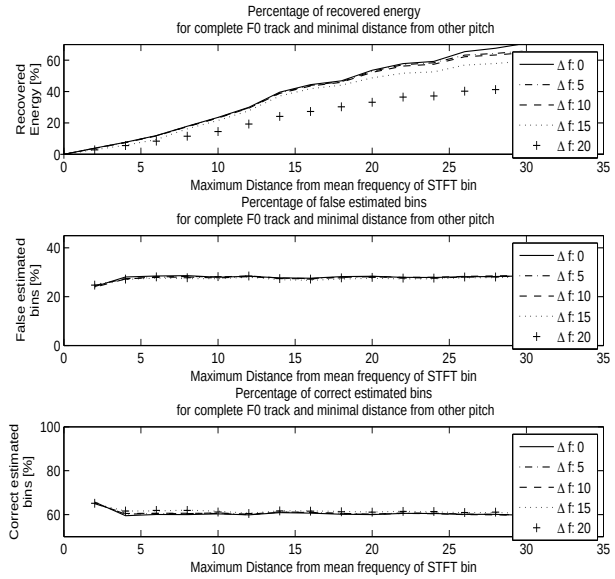


Figure 11: *Performance of the separation algorithm based on known complete F0-track and the distance from the harmonic to the mean frequency of the current STFT bin (Algorithm 6). Shown are several curves for different frequency distances of the nearest interfering harmonic of the interfering speaker.*

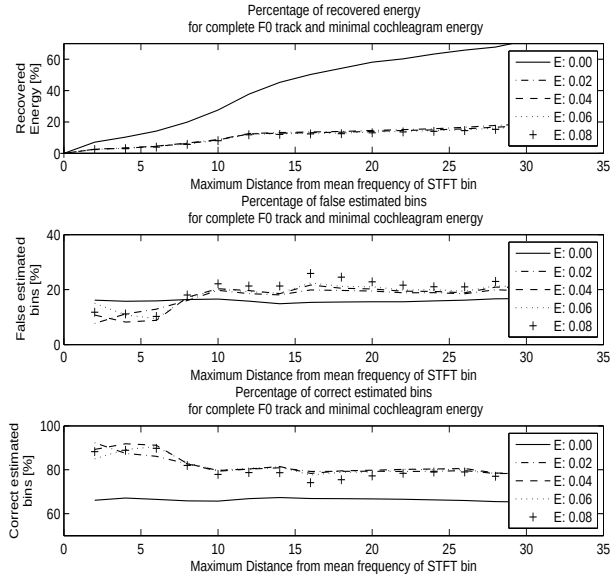


Figure 12: *Performance of separation algorithm based on known complete F0 track and the distance from the harmonic to the mean frequency of the current STFT bin (Algorithm 7). Shown are several curves for different cochleagram energy levels E.*

Algorithm Order	Recovered Energy of ideal mask [%]	Energy of estimated mask belonging to Interferer [%]
$2 \cap 6 \cup 4 \cup 3$	79.89	7.99
$6 \cap 5 \cup 7 \cup 3 \cup 2$	83.90	9.39
$1 \cup 6 \cup 7 \cup 5$	89.09	10.98
$4 \cup 1 \cup 7 \cup 2 \cup 6$	89.82	10.95

Table 1: *A selection of the best evaluation results of the sequential and parallel combining of algorithms 1-7 regarding the percentage of reconstructed energy for an auditory scene consisting of two sources.*

mask for the source of interest. A second central combining stage combines the single masks resulting in a final estimate of the ideal binary mask which is then used to demix the preferred source from the mixture.

A first separation approach combines the estimated masks in a sequential way similar to a chain of responsibility. The first algorithm in the chain assigns all bins according to its specification and passes the remainder of the bins to the second algorithm which in turn assigns those bins that match its specifications and passes the rest to the next algorithm and so on. This sequential combining of the algorithms is equivalent to computing the logical 'or' of the estimated single masks.

To further enhance the final estimated mask, a second approach additionally uses parallel combining to enhance the estimated masks. If several of the algorithms have assigned a specific bin to the preferred source, then this bin is more probable to belong to the source of interest than bins that are only assigned by a single mask. This parallel combining is realized using the logical 'and' of the single estimated masks.

Some results of the evaluation of the combining are summarized in table 1 and 2. The values are obtained by averaging over several recorded mixtures consisting of a female and male speaker positioned at 0° and 45° to the right. The English speech recordings are taken from the CMU speech database [10] and played back at the corresponding directions in a normal office room of size 10×6 m and $RT_{60} = 0.4$ s. The maximum allowed deviation in degree from zero for algorithm 1 and 2 is set to 8° with a minimum correlation peak height of 0.001. The deviation of the STFT phase values used in algorithm 3 is bounded by 11°. Algorithms 5 and 7 use only TF-bins with energy higher than 0.01.

The resulting estimated masks are evaluated by noting mainly two values: The percentage of recovered energy of the preferred source declares how much of the total energy of the ideal mask of the preferred source is reconstructed. A value of 100% states that the estimated mask fully contains the ideal mask. The percentage of interference energy indicates how much energy of the estimated mask belongs to the interfering sources and noise and so is falsely assigned to the estimated mask.

The best estimated mask in terms of percentage of recovered energy is – amongst other combinations not shown for purposes of clarity – calculated using the sequential combination of algorithms 4,1,7,2 and 6. The estimated mask recovers 89.82% of the energy of the preferred source. On the other hand 10.95% of the energy

Algorithm Order	Recovered Energy of ideal mask [%]	Energy of estimated mask belonging to Interferer [%]
$3 \cup 2 \cap 1 \cup 6$	24.48	0.24
$2 \cap 3 \cup 6 \cap 7 \cup 5$	39.36	1.48
$2 \cap 3 \cup 6 \cap 5 \cup 1$	39.83	1.63
$3 \cup 4 \cap 6$	40.23	2.79

Table 2: *A selection of the best evaluation results of the sequential and parallel combining of algorithms 1-7 regarding the percentage of interfering energy for an auditory scene consisting of two sources.*

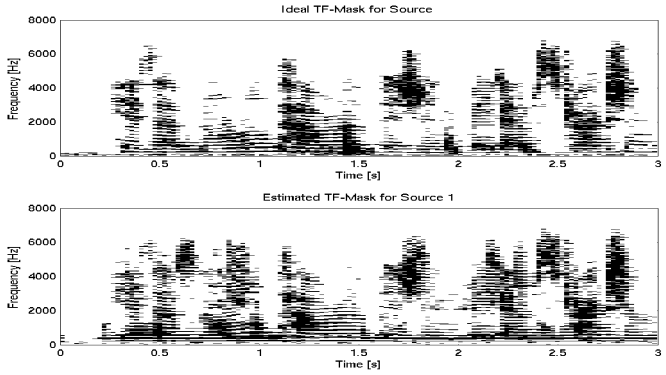


Figure 13: *The ideal mask for the source of interest and the best estimated mask regarding percentage of recovered energy. The recovered energy is* 89.82% *and* 10.95% *of the total energy belongs to the interferers.*

of the estimated mask belongs to the interfering source, yielding a value of 89.05% correct estimated energy. Listening tests result in a very good intelligibility of the source of interest, but the interfering source can be recognized as additional, but very quiet and unintelligible voice in the background.

Table 2 shows a selection of the best estimated masks regarding a minimum energy of interfering sources. Using for example the combination of algorithms $3 \cup 2 \cap 1 \cup 6$ yields estimated masks that recover 24.48% of the total energy of the preferred source while only 0.24% of the total energy of the estimated mask belong to the other source. The intelligibility of the separated speech is quite good and no interfering sources are audible. But compared to the demixed sources of table 1 the reconstructed speech is not so rich and authentic.

The strategy used for combining the masks estimated by the algorithms is dependent on the purpose of the separation infrastructure. If the source of interest is to be enhanced for better intelligibility by humans, sequential strategies should be applied. If however the framework is used as input to an automatic speech recognizer – which in most cases is very sensitive to interfering speech sources – hybrid schemes combining the parallel and sequential strategies are adequate. Other purposes could choose a combination which balances the percentage of recovered and interfering energy to gain an intermediate quality.

7. CONCLUSIONS AND FUTURE WORK

The binaural source separation architecture presented in this paper works well in non-ideal reverberant environments. Prior information regarding the auditory scene are useful to enhance the separation process. Parallel processing paths ensure that the assignment process is optimized regarding the available information at the decision process. By this means the introduced framework achieves quite good separation of speech sources.

Future work especially includes further exploration of the auditory scene. If the source separation algorithms know more characteristics such as the positions of the interfering sources and the respective fundamental frequencies in case of speech sources, the separation could be further enhanced. Additionally in mixed auditory scenes consisting of speech and artificial sources a classification and characterization of each source could assist the separation process.

On the other hand the combining of the masks is currently very rudimentary. Applying higher order inference to the estimated masks will probably further increase the source separation capabilities. Fuzzy logic systems for example can model human reasoning strategies very well and could be applied to infer the dedicated source of each STFT-bin based on all available information.

8. REFERENCES

[1] Ö. Yilmaz and S. Rickard, "Blind separation of speech mixtures via time-frequency masking," *IEEE Transactions on Signal Processing*, vol. 52, no. 7, pp. 1830 – 1847, July 2004.

[2] D. L. Wang, "On ideal binary masks as the computational goal of auditory scene analysis," *In Divenyi P. (ed.), Speech Separation by Humans and Machines*, pp. 181 – 197, 2005.

[3] D. S. Brungart, P. S. Chang, B. D. Simpson, and D.L. Wang, "Isolating the energetic component of speech-on-speech masking with ideal time-frequency segregation," *The Journal of the Acoustical Society of America*, vol. 120, pp. 4007 – 4018, 2006.

[4] D. L. Wang and Guy J. Brown, *Computational Auditory Scene Analysis - Principles, Algorithms, Applications*, IEEE Press, Wiley Interscience, 2006.

[5] G.J Brown and M. P. Cooke, "Computational auditory scene analysis," *Computer speech and language*, vol. 8, pp. 297 – 336, 1994.

[6] Ulrich Reimers, *Digital Video Broadcasting - The Family of International Standards for Digital Video Broadcasting*, Springer, 2005.

[7] Eric Haschke, "Sound source localization using a movable human dummy head," M.S. thesis, Saarland University, 2007.

[8] David Talkin, "A robust algorithm for pitch tracking," *Speech Coding and Synthesis*, pp. 495 – 518, 1995.

[9] Christof Faller and Juha Merimaa, "Source localization in complex listening situations: Selection of binaural cues based on interaural coherence," *The Journal of the Acoustical Society of America*, vol. 116, no. 5, pp. 3075–3089, 2004.

[10] John Kominek and Alan W Black, "CMU ARCTIC databases for speech synthesis," 2003.

SPATIAL TRACK TRANSITION EFFECTS FOR HEADPHONE LISTENING

Aki Härmä and Steven van de Par

Philips Research, Eindhoven, The Netherlands
aki.harma@philips.com

ABSTRACT

In this paper we study the use of different spatial processing techniques to create audio effects for forced transitions between music tracks in headphone listening. The audio effect encompasses a movement of the initially playing track to the side of the listener while the next track to be played moves into a central position simultaneously. We compare seven different methods for creating this effect in a listening test where the task of the user is to characterize the span of the spatial movement of audio play list items around the listener's head. The methods used range from amplitude panning up to full Head Related Transfer Function (HRTF) rendering. It is found that a computationally efficient method using time-varying interaural time differences is equally effective in creating a large spatial span as the full HRTF rendering method.

1. INTRODUCTION

What users commonly do when listening to an audio play list or CD is to jump from one item to another item by pressing the 'Next', or 'Previous' button of the player. This may be performed anywhere between the start and the end of an item and it is implemented in basically all audio players is that the current item is muted and the new track starts playing.

In this paper we study a class of spatial transition effects for headphone listening. The goal is to produce the impression that one track goes physically away and another track comes in. For example, the current music track moves far away to the right and another track slides in from the left hand side. This type of effect has earlier been proposed for surround audio playback with loudspeakers [1] but, to our knowledge, not for headphone listening.

The approach is to position the audio source into a simulated loudspeaker-listener scenario where the virtual loudspeaker, and the listener's ears have well-defined geometric positions. Once this is done, we can move the virtual loudspeaker to arbitrary positions resulting in a perceived movement of the audio sources. In swapping from one audio item to another, the simulation can be performed such that a virtual loudspeaker playing Item 1 is moved far to the left from the user's ears and another loudspeaker playing Item 2 is carried in from the right to the desired playback position. For simplicity, we consider only monophonic audio material in this paper, but the same approach can be used also for stereo or multichannel material by creating multiple virtual loudspeakers.

These effects can be created combining many different methods such as amplitude, or phase panning, HRTF filtering, or room simulation. In the current paper we introduce seven different combinations of algorithms for spatial track transition which differ in computational complexity. Using a new type of listening test, where the subject indicates the movement trajectories of the audio items, we evaluate the effectiveness of each of the methods in creating a large span of perceived auditory movement.

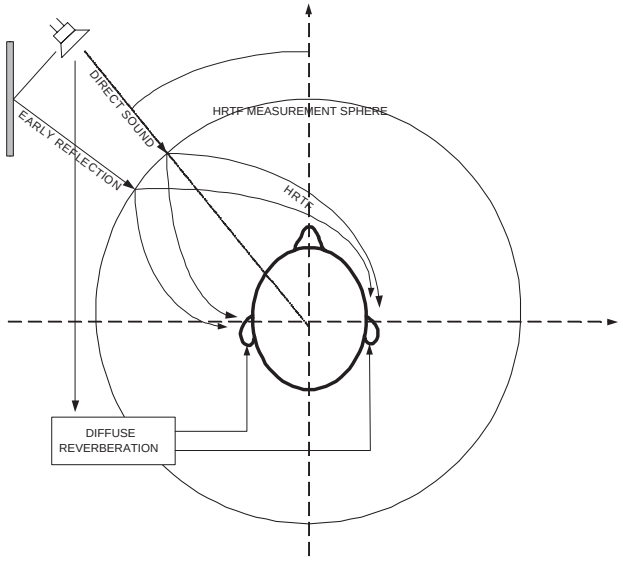


Figure 1: A simulation for a loudspeaker-listener system.

2. THE ACOUSTIC MODEL

The generic model is based on the source-medium-receiver model of binaural simulation [2]. Here, the source is represented by a virtual loudspeaker, the medium is a model of room acoustics, and the receiver is a pair of virtual microphones representing the listener's ears in the room. The model for the medium contains the sound propagation in the room, and the receiver model takes into account the orientation of the listeners head and the Head Related Transfer Functions (HRTFs). This signal processing model is illustrated in Fig. 1 and is similar to those presented by many authors earlier for binaural or transaural listening, see e.g., [3, 2, 4].

The directionality of the source has not been incorporated into the current model, but we assume that the source is an ideal omnidirectional loudspeaker.

In reality, the head-related transfer functions (HRTF) represent impulse responses measured from a limited number of source positions on a sphere with the center position in the center of the listener's head. For example, in the CIPIC data used in the current paper, the radius of the measurement sphere (a hoop where the speakers were fixed) was one meter [5]. The HRTF measurement sphere is shown in Fig. 1. Consequently, the model for a direct sound from a source is a cascade of a direct path filter from the source location to the surface of that sphere, and a HRTF filter from the sphere to the listener's ears. In the room model a limited

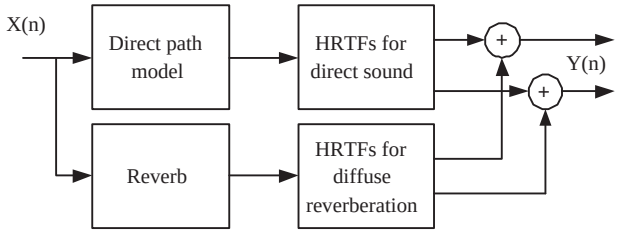


Figure 2: A simulation for a loudspeaker-listener system.

number of early reflections from the room surfaces are often added and they are convolved using HRTFs representing the angles of arrival for each individual reflection, see, e.g, [6]. Finally, the diffuse reverberation is modelled using a filter representing the reverberation and a pair of HRTFs representing the diffuse-field responses to the two ears, that is, means of HRTFs in the horizontal plane.

In this paper, we consider a simplified model consisting only of the direct path and the diffuse reverberation path. The block diagram is illustrated in Fig. 2.

The simulation of a moving sound source can be directly implemented using the system of Fig. 2. For example, the Doppler effect results automatically from changing the delay of the direct path propagation filter as a function of the simulated location of the source. In models for moving sources where the propagation delay has not been implemented, the Doppler effect is sometimes implemented as a separate computational operation, such as frequency modulation in [7] or pitch shifting, to allow better control over the effect.

A signal $x(t)$ played from a virtual loudspeaker is captured using a virtual microphone on the HRTF sphere. The direct sound signal before HRTF filtering is then given by:

$$y_0(t) = x(t,d) * \frac{\delta(t-T)}{d}, \tag{1}$$

where the asterisk denotes convolution, δ is the Dirac's function, $T = d/c$, where c is the speed of sound, d is the distance between the source and its nearest point on the HRTF surface.

For notational convenience we move to the frequency-domain representation of (1):

$$Y_0(\omega) = X(\omega,d)F(\omega,d), \tag{2}$$

where the capital letters denote the Fourier transforms of the parts of (1) and $F(\omega,d) = e^{-i\omega T}d^{-1}$.

Combining all paths from Fig. 2 we may write the synthesis formula (1) in the following form:

$$\begin{bmatrix} Y_l(\omega) \\ Y_r(\omega) \end{bmatrix} = X(\omega,d)\left(F(\omega,d)\begin{vmatrix} H_l(\omega,\alpha) \\ H_r(\omega,\alpha) \end{vmatrix} + \begin{vmatrix} R_l(\omega) \\ R_r(\omega) \end{vmatrix}\right), \tag{3}$$

where $R_l(\omega)$ and $R_r(\omega)$ are the HRTFs to the left and right ear of the listener, respectively, and which depend only on the angle of arrival of the sound α. The model of the reverberation has been integrated with the diffuse field HRTFs into filters $R_l(\omega)$ and $R_r(\omega)$, which are then independent of the source position. In a compact matrix notation we may write this in the following form:

$$\mathbf{Y}(\omega) = X(\omega,d)\mathbf{G}(\omega,\alpha). \tag{4}$$

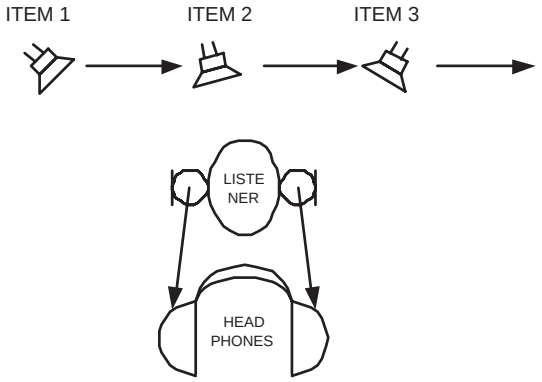


Figure 3: A simulation of a spatial track transition.

3. TRACK TRANSITION EFFECTS

The basic track transition effect studied in the current paper is illustrated in Fig. 3, where virtual loudspeakers representing different audio items that flow past the user. The simulation produces typical spatial audio cues and additionally, due to the direct path delay operator d, a Doppler effect, which is expected to contribute to the perceived illusion of the movement of a source.

4. EXPERIMENTAL SETTINGS

The model of Fig. 2 is computationally expensive mostly due to the HRTF filtering. In a dynamic transition effect, the filter coefficients need to be continuously updated to follow the angle α of the sound source. In practice this requires continuous interpolation of the responses to reduce artifacts related to switching filters. The model of the reverberation is another expensive part because it typically requires implementation of a high-order FIR filter.

In this paper, we study seven different systems that differ in the degree to which simplifications have been made to the signal processing model. They are all studied in the configuration illustrated in Fig. 3, where the monophonic sources move along a line from the left to the right such that, when allowed by the method, sources pass the listener at a constant speed at $\alpha = 0^o$ at the distance of $r = 2$ meters from the listener. The block diagrams of the methods are shown in Fig. 4.

In the pure amplitude panning method (a) the gains for the two ear signals are given by:

$$\mathbf{G}_a(\omega) = \begin{vmatrix} g_n \\ g_0 g_n \end{vmatrix}, \tag{5}$$

where:

$$g_0 = 10^{\frac{14}{40\pi}\mathrm{atan}(p(n)/r)} \text{ and } g_n = \frac{1}{\sqrt{1+g_0^2}}, \tag{6}$$

where $p(n)$ is position of the sound source as a function of the sampling number n. The equation approximates the listening test data on binaural lateralization of a source in dichotic listening with only level differences between the two ears [4].

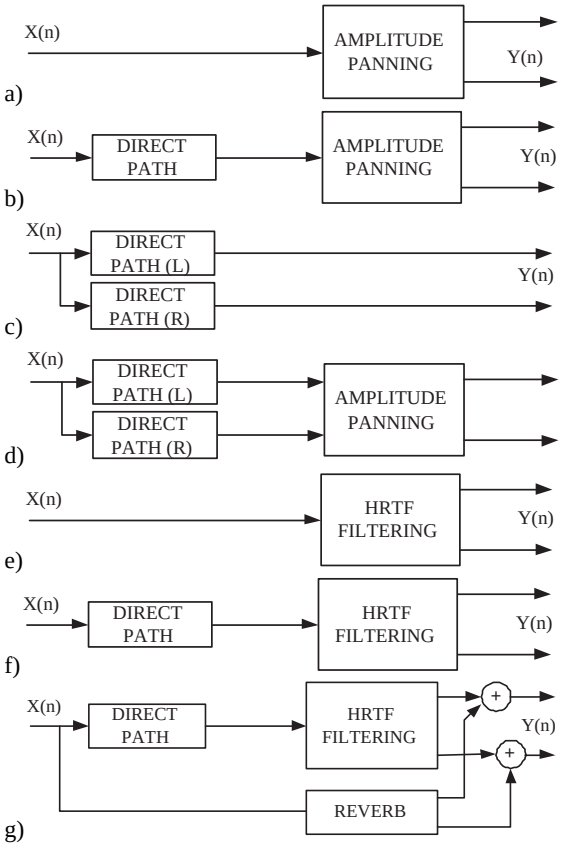


Figure 4: The models for spatial track transition studied in this paper.

The second model (b) combines the amplitude panning with a single direct path model. It can be written in the following form:

$$\mathbf{G}_b(\omega, n) = F(\omega, \sqrt{r^2 + p^2(n)}) \left| \begin{array}{c} g_n \\ g_0 g_n \end{array} \right|, \qquad (7)$$

where the only difference to (5) is the direct path model $F(\cdot)$, which in the case where $x(n)$ changes over time produces a Doppler effect. The fractional delays were implemented in the time domain using the sixth-order Lagrange FIR interpolator. The third model (c) is essentially a free-field model for a listener with an acoustically transparent head. The synthesis formula is given by:

$$\mathbf{G}_c(\omega, n) = \left| \begin{array}{c} F(\omega, \sqrt{r^2 + (p(n) + h/2)^2}) \\ F(\omega, \sqrt{r^2 + (p(n) - h/2)^2}) \end{array} \right|, \qquad (8)$$

where h is the distance between the two ears of the listener. In our simulations we used $h = 0.2$ m, which is somewhat larger than the human average.

The next model (d) is obtained by including a very simple model for the head shadowing to model (c). In fact, the head-shadowing model is exactly the same as the binaural amplitude panning used in (a)-(b), and it yields:

$$\mathbf{G}_d(\omega, n) = \left| \begin{array}{c} g_n F(\omega, \sqrt{r^2 + (p(n) + h/2)^2}) \\ g_0 g_n F(\omega, \sqrt{r^2 + (p(n) - h/2)^2}) \end{array} \right|. \qquad (9)$$

Note, that this model implements an approximative HRTF model, producing the same ITD and ILD cues at all frequencies with one delay and one gain coefficient per channel.

In model (e) we use the HRTF data from the CIPIC database (subject 31) [5]. The azimuthal set of HRTFs at the frontal area was augmented by a number of interpolated HRTF impulse responses. The interpolation was performed linearly in the frequency domain separately for magnitude and unwrapped phase responses. The synthesis equation is given by:

$$\mathbf{G}_e(\omega, n) = \left| \begin{array}{c} H_l(\omega, \mathrm{atan}(p(n)/r)) \\ H_r(\omega, \mathrm{atan}(p(n)/r)) \end{array} \right|, \qquad (10)$$

and the convolutions were implemented efficiently using the FFT overlap-add techniques.

The next model (f) incorporates the direct path model to the HRTF model:

$$\mathbf{G}_f(\omega, n) = F(\omega, \sqrt{p^2(n) + r^2} - r_{\mathrm{hrtf}}) \mathbf{G}_e(\omega, n), \qquad (11)$$

where $r_{\mathrm{hrtf}} = 2$ m is the radius of the HRTF measurement sphere of Fig. 2.

Finally, model (g) includes a model of the diffuse reverberation:

$$\mathbf{G}_g(\omega, n) = \mathbf{G}_f(\omega, n) + \left| \begin{array}{c} R_l(\omega) \\ R_r(\omega) \end{array} \right|, \qquad (12)$$

where the filters $R_l(\omega)$ and $R_r(\omega)$ are synthetic pink noise sequences with the temporal envelope from a real room impulse response with the reverberation time of $T_{60} = 1.0$ s.

5. LISTENING TEST

The purpose of the algorithms introduced above is to provide a spatial experience of a movement of an audio source in headphone listening. Generally, the hearing mechanism does not seem to be sensitive to the movement itself [8]. It is often suggested that the percept of the movement is a consequence of observing the source first at one position, and then at another position. However, there is evidence on brain areas that are actually sensitive to the movement of sound sources [9].

The just noticeable difference for the velocity of a source is typically in the range of 4-9 degrees per second [9] or 1.5 to 4.6 m/s [10] for a source moving a linear trajectory 5 meters in front of the listener. In most studied on just-noticeable differences (jnds) of velocity perception [11, 10] it has been found that the most important cues for the velocity are the Doppler effect and the changes in the overall loudness. The binaural cues including interaural time (ILD) and level differences (ILD) are weaker cues in the velocity discrimination. However, in another experiment where the listeners' task was to indicate the point where a moving source is closest to the listener suggested overall loudness to be the most important cue, followed by dynamic ITD cue, and only then the Doppler effect [12]. In the current article, the primary goal is to create an illusion of a large movement with a low-complexity algorithm, therefore the velocity or the temporal position of a source are not necessarily as important as the perceived distance the source has travelled, that is, the range of the movement.

In this paper, we developed a listening test that aims at depicting the subjective experience of a movement of a source in the case where the user is *scanning* over a sequence of three consecutive samples in a playlist of audio items. A similar movement pattern where the sound source moves along a linear horizontal trajectory

1.5 meters in front of the listener was used in all methods. The movement pattern was such that each new source appeared at the distance of 20 meters to the right of the listener moving to the front of the listener in two seconds, stopping at the front for one second, and then moving in two seconds 20 meters to the left. In methods where the rendering depends only on the angle of the source, such as amplitude panning and pure HRTF rendering, only the angle derived from the position of the source was used. Since the overall loudness has been found to be a dominating cue in listening experiments with moving sources [12, 11, 9, 10] an identical amplitude weighting as a function of the position was used with all the methods.

The test was performed in a sound insulated booth using Beyerdynamic DT990 headphones. Ten subjects participated in the experiment. The test material consisted of five playlists of three audio items each. Three of the playlists represented three-second excerpts from samples of different music genres (rock, pop, rap), one playlist consisted of uncorrelated pink noise sequences, and finally one playlist had rich harmonic tone complexes at three different fundamental frequencies. In the listening test, subjects were asked to listen to a sequence and then draw, using the computer mouse, their subjective impression of the path of the sequence of three audio items on a chart illustrated in Fig. 5X). We projected each drawing to a bitmap of 40×40 pixels for analysis.

6. RESULTS

To compare the span of path assessments in the different methods we computed pixel-wise 2D histograms over all listeners and play lists. The 2D histograms for the seven methods are shown in Figs. 5a-g. The figures show that there are differences between the subjective assessments of the transition paths in the methods discussed in this paper. In Method (a), the path is mainly judged inside the listener's head, while in other methods the span of the effect appears larger. The marginal histogram plotted at the bottom of each panel also suggests that the histograms are tilted towards the right, even if the movement path was symmetrical from the left to the right. This is an interesting finding.

It was found that there are significant differences between individual listeners. The differences are probably largely due to the differences in the ways how individual subjects mapped the auditory experience to a visual geometric form.

In order not to be influenced by these individual differences we decided to convert the results to a relative scale with a pairwise comparison of the individual path drawings for the different rendering methods. Table 1 gives the percentage for the probability that a rightmost point in a path in method X (row) is farther to the right than the corresponding point in the path for method Y (column) in one listener. Comparing methods (a) and (b) in Table 1, we see that the percentages are almost 50%, which means that the methods are essentially similar in the span to the right hand side. The percentages that the path spans farther to the right in methods c-g is 78-94% over the methods a-b. The method (c) gives a higher percentage over method a) than method (b) does. Both methods (b) and (c) contain the distance model. In method (b) the effect causes only the Doppler effect, while method (c) creates an interaural time difference which changes dynamically over the transition path.

It is interesting to note that a computationally light method (d) combining dynamic interaural time difference (with a transparent head model) and amplitude panning gets very similar rankings

A/B|	a	b	c	d	e	f	g
a	0	48	78	82	96	88	82
b	52	0	82	78	84	92	84
c	16	18	0	56	78	72	56
d	14	22	40	0	78	68	48
e	4	14	22	22	0	30	28
f	10	8	28	32	66	0	40
g	18	16	42	50	72	58	0

Table 1: Probability that path produced using transition A has a larger span to the *right* than transition B.

with the most complex method (g). The results suggest that the pure HRTF rendering (method e) gives systematically the largest spatial span to the right. In fact, methods (f) and (g), which add the dynamic distance model, and diffuse reverberation to the pure HRTF model get lower gradings than model (e).

Table 2 gives the percentages for the leftmost point in the path. The results support the observation that the span of the path in the amplitude panning models (a-b) is smaller than in the other methods with interaural time difference cues. However, the percentages are now lower. Comparing the method (a) to method (b), and (e) to (f) suggests that the use of the one-channel distance effect in the form of a Doppler effect in fact decreases the span of the path to the left.

A/B|	a	b	c	d	e	f	g
a	0	40	58	72	74	66	58
b	58	0	72	74	72	72	60
c	40	24	0	68	68	68	60
d	26	26	32	0	62	54	46
e	24	20	30	34	0	44	40
f	32	20	28	44	54	0	42
g	38	30	40	50	52	54	0

Table 2: Probability that path produced using transition A has a larger span to the *left* than transition B.

In the frontal area, see Table 3, it seems that the pure HRTF rendering again gives the largest span to the front. The low score of method (d) in the frontal area is an unexpected result because it gives a large range in right-left direction.

A/B|	a	b	c	d	e	f	g
a	0	60	62	46	66	62	60
b	32	0	56	44	60	44	70
c	34	38	0	38	54	40	58
d	38	52	56	0	60	54	64
e	28	38	36	28	0	42	48
f	34	46	42	34	52	0	60
g	26	26	36	30	40	38	0

Table 3: Probability that path produced using transition A has a larger span to the *front* than transition B.

The intended path passed the users face 1.5 meters at the front of the listener. However, several listeners papered a path behind

the head, too. The comparison in Table 4 suggests that the localization at the back of the head, or behind the head was strongest in amplitude panning methods (a-b), and somewhat increase also in method (d). For example, in 66% of the cases the path drawn for the method (d) span to the back more than the path for the same playlist in method (e).

A/B|	a	b	c	d	e	f	g
a	0	48	30	34	24	24	30
b	46	0	22	28	16	26	22
c	62	72	0	60	32	44	46
d	58	68	40	0	28	40	38
e	64	76	62	66	0	50	56
f	66	66	46	50	38	0	44
g	66	70	46	54	32	48	0

Table 4: Probability that path produced using transition A has a larger span to the *back* than transition B.

7. CONCLUSIONS

In this paper we have studied seven different techniques for the dynamic rendering of sound sources in spatial track transition. In particular, we have focused on a track transition effect where one song comes from the left hand side of the user and disappears to the right. The techniques represent different levels of computational complexity. The simplest techniques are based on dynamic amplitude panning, that is, multiplication of the signal with a scalar coefficient. The most complicated reference method combines HRTF filtering, the Doppler effect, and room reverberation.

The listening tests suggest that the amplitude panning techniques give generally a narrow range of the effect and the image is often inside the head. The plain HRTF processing gives the largest span in the lateral direction. However, a simple method combining delay panning and amplitude panning appears almost equally powerful for the creation of left-right transition effects. However, the HRTF method appears giving a slightly larger span in the front left direction and possibly better externalization.

The addition of room reverberation produced an interesting effect but the benefits are not obvious in the current results. It appears that for most listeners the method combining HRTF filtering and reverberation gave smaller left-right span that the pure HRTF filtering.

In the listening tests the goal was to compare the different methods by the perceived spatial span of the transition effect. This is a different task from the subjective evaluation of the velocity of a source [11, 9, 10] or the closest point in the movement trajectory [12]. In velocity discrimination studies the Doppler effect and the overall loudness have been found to be more important than the binaural cues such as ILD or ITD. The spatial delay panning methods aim at creating a distance cue. In all cases the transition effect was tuned in such a way that it created an audible Doppler effect during the transition of an audio playlist item from left to the right. The Doppler effect was audible in five out of the seven methods. If was found that when the Doppler effect appeared without associated binaural time difference cues, it had almost no influence on the perceived left-right span. In particular, the difference between amplitude panning with and without the Doppler effect was small but the difference between the amplitude panning with the Doppler effect, and the method where amplitude panning was combined with time-varying interaural time-differences was significant. The results seem to suggest that the interaural time-differences actually play a more important role in the perceived span of a transition than the Doppler effect.

From the results of the current listening test we may conclude that the dynamics of interaural time-differences are important in producing a large spatial span for the track transition effects. In addition, a very simple method based on a computationally very efficient simplified sound propagation model to the two ears of a listener gives almost equally good results in the span of the movement effect as a more complicated method based on the measured head-related transfer functions.

8. ACKNOWLEDGEMENTS

The authors are grateful to Armin Kohlrausch for fruitful discussions related to the design of the test setup, Bert den Brinker and Othmar Schimmel for help in improving the manuscript, and finally the listening test subjects for their fine illustrations of auditory motion patterns.

9. REFERENCES

[1] T. Herberger and T. Tost, "System and method for generating sound transitions in a surround environment." US Patent 2005/0047614A1, 2005.

[2] L. Savioja, J. Huopaniemi, T. Lokki, and R. Väänänen, "Creating interactive virtual acoustic environments," *J. Audio Eng. Soc.*, vol. 47, no. 9, pp. 675–705, 1999.

[3] D. R. Begault, *3-D sound for virtual reality and multimedia*. New York, USA: Academic Press, 1994.

[4] J. Blauert, *Spatial Hearing: The Psychophysics of Human Sound Localization*. Cambridge, MA, USA: The MIT Press, 1999.

[5] V. R. Algazi, R. O. Duda, D. M. Thompson, and C. Avendano, "The CIPIC HRTF database," in *Proc. IEEE WASPAA'2001*, (New Paltz, NY, USA), October 2001.

[6] J.-M. Jot, "Scene description model and rendering engine for interactive virtual acoustics," in *AES 120th Conv. preprint 6660*, (Paris, France), May 2006.

[7] J. M. Chowning, "The simulation of moving sound sources," *J. Audio Eng. Soc.*, vol. 19, pp. 2–6, January 1971.

[8] D. W. Grantham, *Hearing (ed. B. J. C. Moore)*, ch. Spatial hearing and related phenomena, pp. 297–339. Academic Press, 1995.

[9] S. Carlile and V. Best, "Discrimination of sound source velocity in human listeners," *J. Acoust. Soc. Am.*, vol. 111, pp. 1026–1035, February 2002.

[10] T. Kaczmarek, "Auditory perception of sound source velocity," *J. Acoust. Soc. Am.*, vol. 117, pp. 3149–3156, May 2005.

[11] R. A. Lutfi and W. Wang, "Correlation analysis of acoustic cues for the discrimination of auditory motion," *J. Acoust. Soc. Am.*, vol. 106, pp. 919–928, August 1999.

[12] L. D. Rosenblum, C. Carello, and R. E. Pastore, "Relative effectiveness of three stimulus variables for locating a moving sound source," *Perception*, vol. 16, pp. 175–186, 1987.

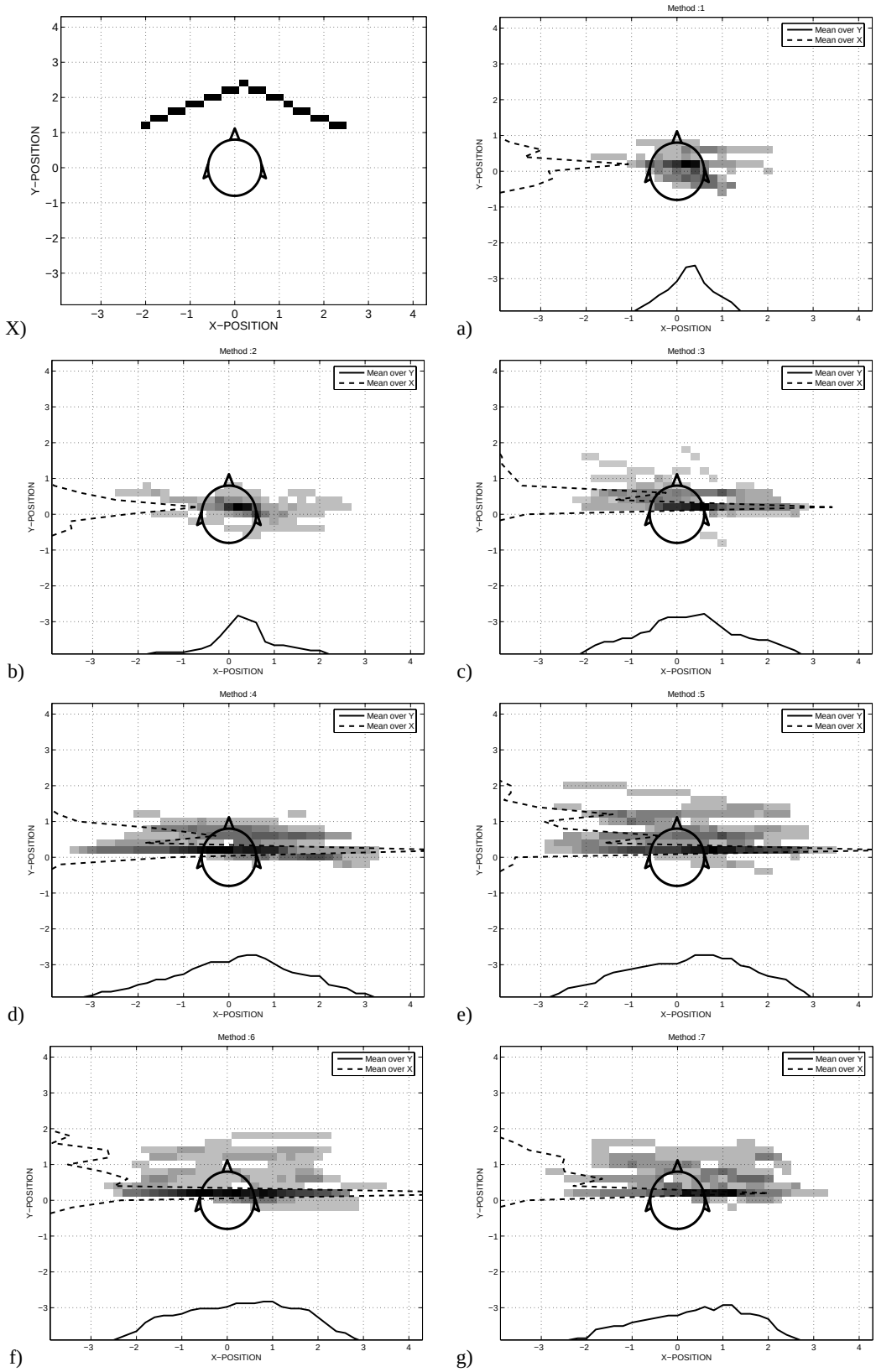


Figure 5: X) An example of a drawing of a user for one playlist and rendering method. a)-g) Histograms of subjective path assessments over all subjects and play lists for all the seven methods. A dark color indicate that the path is often drawn through the pixel.

MONOPHONIC SOURCE LOCALIZATION FOR A DISTRIBUTED AUDIENCE IN A SMALL CONCERT HALL

Enda Bates, Gavin Kearney, Frank Boland and Dermot Furlong

Department of Electronic and Electrical Engineering
Trinity College Dublin, Ireland

batesja@tcd.ie, gpkearney@ee.tcd.ie, fboland@tcd.ie, dermot.furlong@tcd.ie

ABSTRACT

The transfer of multichannel spatialization schemes from the studio to the concert hall presents numerous challenges to the contemporary spatial music composer or engineer. The presence of a reverberant listening environment coupled with a distributed audience are significant factors in the presentation of multichannel spatial music. This paper presents a review of the existing research on the localization performance of various spatialization techniques and their ability to cater for a distributed audience. As the first-step in a major comparative study of such techniques, the results of listening tests for monophonic source localization for a distributed audience in a reverberant space are presented. These results provide a measure of the best possible performance that can be expected from any spatialization technique under similar conditions.

Keywords: Sound localization, distributed audience, spatial music.

1. INTRODUCTION

Much of the existing research into the localization performance of spatialization techniques has been carried out under anechoic conditions for a single listener. While this approach is suitable for evaluating the optimal performance of a particular system, it does not define the capability of these systems in real reverberant concert hall environments. This is particularly relevant in the area of spatial music composition, where the process of transferring spatial locations and trajectories from the studio to the concert hall presents significant challenges. The two main factors which are particularly relevant to this issue are, early reflections and reverberation, and an extended listening area. Research is required to gauge the performance of different spatialization schemes for a distributed audience and in reverberant concert halls.

However, before any assessment of multichannel spatialization techniques can be made, it is necessary to examine the performance of monophonic sources under the same conditions. In this paper we present the results of listening tests carried out in a small sized concert hall for a distributed audience of nine people. The tests were conducted using a monophonic loudspeaker array with various stimuli and illustrate the impact of reverberation on the localization performance of a distributed audience for single sources. This will provide a base measure of the best possible performance one could expect from a spatialization technique in terms of localization accuracy, under similar conditions. We will begin this study with a brief overview of the auditory localization mechanisms, and a summary of the existing research and experimental data on source localization.

2. AUDITORY LOCALIZATION OF MONOPHONIC SOURCES

The localization of sound sources can be divided into three spatial categories, namely directional hearing in the horizontal plane, directional hearing in the vertical plane and "distance hearing" [1]. In this study we will limit our discussion to the horizontal plane, as this is particularly relevant for most spatial music presentations. The ability to localize auditory events in this plane depends on several key factors. These include the nature of the source signal, the acoustical environment and the diffraction effects of the upper torso and head. Of these factors, head shadowing gives rise to interaural level differences (ILD) and ear positioning gives interaural time differences (ITD) which aid our horizontal localization [2]. The implications (and limitations) of these cues in the free field have been well documented in [1, 3, 4, 5] in terms of the nature of the source. There is a strong weighting for ITDs with low frequency signals and poor weighting of ITDs with high frequency signals. The converse is true for the ILD. For wideband stimuli, the ITD is found to dominate [3].

It has been shown under ideal conditions, for various source stimuli, that the region of most precise spatial hearing lies in the forward direction with frontal hearing having an accuracy of between 4.4° and 10° for most signal types [1]. Localization ability decreases as the source azimuth moves to the sides, with the localization blur at $\pm 90^\circ$ being between three to ten times its value for the forward direction. For sources to the rear of the listener, localization blur improves somewhat but is still approximately twice that for frontal sources. It is expected therefore that the localization performance of spatialization systems will follow a similar trend.

A thorough study on the effect of reverberant conditions on localization accuracy for various stimuli was presented by Hartmann [6, 7, 8]. It was shown that impulsive sounds with strong attack transients are localized independently of the room reverberation time, but may depend on the room geometry. Conversely, for sounds without attack transients, localization improves monotonically with the spectral density of the source. However, localization of *continuous* broadband noise is dependent on room reverberation time.

The source must also include significant onsets if the 'precedence effect' is to operate as an aid to localization. However, even with transients the precedence effect does not entirely eliminate the effect of early reflections [7]. In fact, the early reflections from room sides impact negatively on horizontal localization, while early reflections from the floor and ceiling help to reinforce localization. It should be noted that this is the opposite of the preferred arrangement for acoustic music in concert halls, which

emphasizes lateral reflections.

3. LOCALIZATION OF PHANTOM SOURCES

In assessing which systems are most applicable to the presentation of spatial music, it is relevant to discuss the development of commercial sound reinforcement and spatialization systems that are applicable to localization of phantom sources. The first of these is stereophonic reproduction, which refers to the creation of virtual acoustic images localized at a desired position. The original patent by Blumlein [9] in 1931 outlines two-channel stereophony around a single listener position and remains the main commercial system of sound reproduction to this day. However, presentations using this system are designed to provide accurate imaging for a single listener position only. Off-centre listening leads to inaccurate localization information since the intensity information presented to the ears becomes compromised. Three channel stereophony attempts to overcome this, but again, it does not cater for large deviations from the acoustic 'sweet spot' and does not provide any accurate localization information for anything other than the frontal plane.

Ville Pulkki [10] created a vector-based reformulation of the amplitude panning method (VBAP) which extends the basic stereophonic principle to an arbitrary number of loudspeakers. A number of experiments were undertaken using this system to investigate the perceptual cues used in source localization, both for real and phantom sources. In [11] the localization of amplitude panned phantom sources in a standard stereophonic system was investigated. Of particular interest is his use of a binaural auditory model to calculate the localization cues for the audio signals used in the listening tests. These data simulations were compared with the results of the perceptual listening tests in order to verify the experimental results. It should be noted, however, that this model does not take into account the precedence effect and only gives reliable results if the sound signal arrives at the ears within a 1-ms window. The experiment was therefore carried out under anechoic conditions and a modified model would be required for tests under reverberant conditions. The results of this experiment correlate well with Blauert [1] for the following points:

- The localization of amplitude panned phantom sources is based on ITD cues at low frequencies and on ILD cues at high frequencies.
- ILD cues at high frequencies generally coincide with low-frequency ITD cues.
- Between 1100 and 2600 Hz both cues become ambiguous.

These results explain why broadband phantom sources are generally well localized while narrowband signals, particularly in the region of 1.7kHz, are inconsistently localized.

Ambisonics is another system which surpasses the limits imposed by two channel stereophony and is a very complete set of techniques for recording, manipulating and synthesizing artificial sound fields [12]. Real soundfields can be recorded using a specialized Soundfield microphone, while numerous software implementations allow for the synthesis of artificial sound fields. Ambisonics has been widely discussed and excellent overviews can be found in [13, 12]. One of the most lauded features of Ambisonics is that the encoding and decoding functions are carried out separately. This capability certainly gives Ambisonics an advantage over systems based on amplitude panning which require a dedicated channel per loudspeaker and a fixed arrangement of loudspeakers. However, while the localization performance of Ambisonic systems has been evaluated in a number of experiments [14, 15], there is a distinct lack of experimental data on its performance under non-anechoic conditions and for a distributed audience. Higher order Ambisonic systems will theoretically recreate the soundfield over a wider listening area but again, additional testing will be required to verify this claim. There is also a lack of consensus amongst practitioners as to the most appropriate decoding equations for different environments.

Benjamin et al. [15] carried out a series of listening tests to verify the various theories behind different Ambisonics decoder designs. The tests compared a number of different speaker arrays and decoder designs, with the main variables being the number and arrangement of the loudspeakers, and the psychoacoustic models guiding the decoder design. The tests were designed to evaluate these models, and the choice of crossover frequency. Three decoders were designed, a velocity decoder in which the original pressure and particle velocity are recovered exactly, an energy decoder that maximizes the magnitude of the energy localization vector, and a shelf decoder which optimizes the velocity vector at low frequencies and the energy vector at mid frequencies. These three configurations were then applied to the decoding equations for square, rectangular and hexagonal arrays to generate the test signals. The source signals used in the test consisted of continuous bandpass filtered noise, voice recordings, various music recordings, applause and fireworks. The listeners were free to switch between the arrays and sources, move their heads and seating position and were asked to judge a number of attributes such as the directional accuracy of localization, tonal balance and image stability. The results of the test indicated that the hexagonal array was preferred by all listeners. The rectangular and square arrays were judged to exhibit poor lateral imaging although the rectangular array was comparable to the hexagonal array when the material was limited to a frontal source with ambience. Of the four decoder types tested, the shelf filter decoder was preferred for most sources as it produced the most focused sources with the least artifacts. One interesting conclusion drawn from this test was that changes in layout make significantly more difference than changes in decoder. Benjamin et al. also note that the choice of preferred decoder was strongly dependent on the program material and the size of the intended listening area. Finally, it should be noted that this test was initially carried out in an ordinary room without any acoustic treatment. It was reported that good localization was not achieved and no experimental data was presented from these initial tests.

It is felt by the authors that these reported studies show that VBAP and Ambisonics are viable formats for the production of spatial music in ideal listening environments. What is unclear, however, is the true capability of such systems to cater for the localization requirements of a distributed audience in a concert hall environment.

4. SOUND LOCALIZATION FOR A DISTRIBUTED AUDIENCE

There are numerous challenges associated with the localization of sources for distributed audiences. In particular, for systems based on stereophonic principles, it is extremely difficult to present accurate wavefronts for correct ITD/ILD cues at off-centre listening po-

sitions. Even for Ambisonics, which has been used extensively in theatre and electro-acoustic music concerts, there has been very little published on the actual performance of the system under these conditions. David Malham has worked on a number of large-area Ambisonics systems and has published one of the few papers on this topic [12]. The paper informally covers the experiences of the author in implementing large scale Ambisonics systems in a number of different theatres. The main conclusions of the paper are as follows:

- Informal tests demonstrated that Ambisonics worked effectively with a hexagonal array of diameter 14.5m.
- Non-central listening positions produced distortions in the sound field positions.
- Audience screening is a significant problem for periphonic, three dimensional presentations.
- It is important to distinguish between imaging problems caused by system faults and those resulting from systematic errors caused by the acoustics of the projection space or the nature of the sound being projected
- Decoding based on the diametrically opposed pairs theorem performs poorly for large arrays and should only be used for small listening areas.
- Fast moving sounds were more easily localized.
- The system can work well even for listeners placed outside the array, but not for listeners seated on the surface of the notional sphere of the loudspeaker array.
- The acoustics of the venue strongly influence the effectiveness of the system.

Another system worthy of mention for distributed audiences is the Delta Stereophony System (DSS) as it prioritises delivery of the correct wavefronts for accurate source localization and boasts true perspective and depth [19]. DSS is largely based on the precedence effect and is an approach which ensures that each listener in an auditorium receives the direct sound from the original sound source direction first, before that of reinforcement speakers placed about the audience area [20]. In that it was intended for sound reinforcement system use in large auditoria, it employs a distributed loudspeaker network, with loudspeakers typically positioned throughout an auditorium. The main objective of DSS is to reinforce an original sound event while also maintaining at least an approximately accurate sound source localization. This can be achieved if the listener at any place in the room receives the first wavefront from the direction of the sound event being reinforced, rather than from any of the other loudspeaker positions [21]. Since the development of DSS in 1975, it has been installed in concert halls in Berlin, Prague, Munich, Stade, Stuttgart, Tokyo, and the Moscow Kremlin Palace [22]. It has also been applied with great success in open air theatres such as the Lake Festival Bregenz in Austria (where it was used for reinforcement of moving sources), Trachselwald (Switzerland), and Waldbuhne, Berlin. Further examples of DSS implementations can be found in [23, 24]. Ahnert gives an excellent review of DSS design in [20], but the actual subjective system performance in terms of localization accuracy in a reverberant environment for a distributed audience, and for differing source material, still has to presented. It is felt by the authors that since the DSS is designed for the accurate localization of sources, it is worthy for inclusion in the assessment of systems applied to spatial music presentations.

In recent years, another spatialization method has been developed by Berkhout et al [16], namely Wave Field Synthesis (WFS). The theoretical background for this system originates in Huygen's principle in optics, where a wavefront can be reconstructed by an infinite series of secondary wavefronts. In practice, the number of secondary sources is limited and the spatial separation between the loudspeakers determines the highest frequency that can be reconstructed accurately. A series of experiments was set up by De Vries et al [17] in an effort to gain experience of a sound enhancement system based on Wave Front Synthesis. They constructed three Wave Front Synthesis systems: a laboratory setup, a prototype system, and a full-sized sound enhancement system. De Vries found that the spatial bandwidth of a single notional source can be approximated by the dispersion angle of the source. Thus, increasing the directivity of the notional source increases the spatial aliasing frequency f_{al}. However, if the spatial bandwidth is reduced too far, this can also cause localization problems for listeners located at the far sides of the rooms. Listening tests in the auditorium confirmed that when the wave front synthesis is aliasing-free up to higher frequencies, the perceived source images are narrower and more accurately localized and coloration effects are reduced.

5. TOWARDS ASSESSMENT OF SPATIAL ENHANCEMENT SYSTEMS

In order to effectively gauge the subjective performance of any spatial enhancement system in a reverberant environment, it is first necessary to study the effect of room acoustics on localization accuracy, in particular for distributed audiences. This can then be considered as the 'best case' scenario for any sound system in the same environment. In light of this, and as a precursor to studies by the authors for testing the localization accuracy of various spatial enhancement systems [25, 26], a series of experiments were set up in a small sized concert hall in Trinity College Dublin. The hall, shown in Figure 1, has a reverberation time (RT60) of 0.9 seconds at 1kHz. A loudspeaker array consisting of 16 Genelec 1029A

Figure 1: Printing House Hall in Trinity College Dublin showing listener/loudspeaker setup.

loudspeakers was arranged around a 9 listener audience area as shown in Figure 2. A PC utilising a MOTU896 audio interface

was used to route the audio to the loudspeakers. The loudspeakers

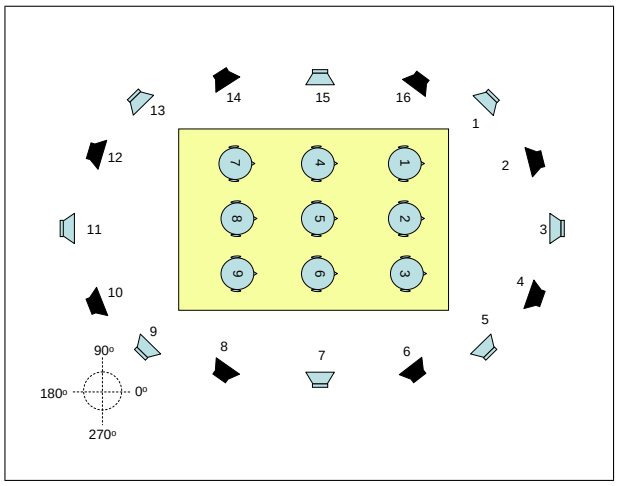


Figure 2: Geometry of loudspeaker array and audience area for monophonic listening tests.

were calibrated to 80dBA at 1m from the on-axis tweeter position and their axis lines were coincident with the centre listener position. The audience, which consisted primarily of students under 35 years of age, were screened before the tests for potential hearing impairments. The participants were presented with monophonic sound from pseudorandom positions located about the speaker array and were then asked to identify the location of the sources via a questionnaire running concurrently with the tests. This randomized method was used to negate any order effects during the tests.

In these tests, only the 8 black loudspeakers shown in Figure 2 were used for the monophonic presentations, and the other 'dummy' loudspeakers were used to increase the choice of angle for the listeners. In order to assess the effect of various stimuli, users were presented with 1 second unfiltered recordings of male speech, female speech, Gaussian white noise and music with fast transients. These samples have the spectral and temporal characteristics shown in Figure 3. Each sample was presented twice, followed by a short interval before the next presentation. Listeners were asked to keep their heads in the forward direction and the angular conventions employed in the analysis at each individual listener position are also shown in Figure 2. Upon completion of one iteration of the test each listener was asked to move to the next seat for another randomised iteration.

Each of the listeners' answers were weighted, depending on the confidence level of the listener with their choice, with weightings of $1/n$, where n is the number (or range) of speakers that a listener felt the sound originated from. From this, the histogram $\{h(\theta_i)\}_{i\in[1:16]}$ collecting all the listeners' answers is computed for each seat. The angular mean $\bar{\theta}$ and the unbiased standard deviation σ_θ at each listener position are computed:

$$\bar{\theta} = \frac{\sum_{i=1}^{16} h(\theta_i)\cdot\theta_i}{\sum_{i=1}^{16} h(\theta_i)} \tag{1}$$

$$\sigma_\theta = \sqrt{\frac{\sum_{i=1}^{16} h(\theta_i)(\theta_i-\bar{\theta})^2}{(\sum_{i=1}^{16} h(\theta_i))-1}} \tag{2}$$

In some rare situations anomalous statistical outliers would occur with large deviations from the data set and actual loudspeaker

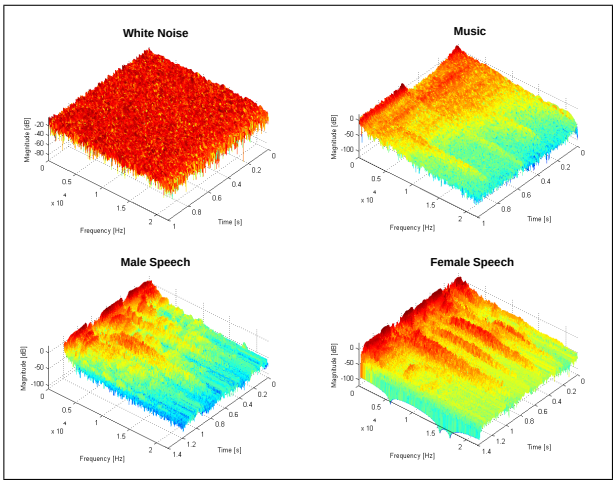


Figure 3: Spectral and temporal characteristics of presented sources.

angle θ_T. Such anomalies were attributed to inattentive listeners or individual listener problems during the tests. These anomalies were removed from the histogram. Consequently in these rare cases, the term $\sum_{i=1}^{16} h(\theta_i)$ becomes less than the number of listeners (i.e. < 9) but is never less than 8.

Figures 4, 5, 6 and 7 show the results of the measurements taken. Each figure contains four graphs indicating the measured localization data for each source signal. Note that the Y-axis limits on each graph set is different to accommodate the resolution at each listener position. The individual plots show the mean $\bar{\theta}$ (circle), twice the deviation $2\sigma_\theta$ (whiskers) and presented localization angle, or ground truth (square) from the perspective of each listener position. Figure 4 shows the results for a frontal presentation from speaker 2. One can note the following:

1. The mean results for all source signals match the presented source angle except for white noise at listening position 6.
2. All sources were localized to the presented location with zero deviation except for white noise with a deviation of $\pm 7.4^\circ$ about the mean at listening position 6.

These results indicate that the localization accuracy of a distributed audience for a frontal source is quite good, and is largely independent of the type of source signal used.

Figure 6 shows the results of a rear presentation from speaker 10. One can note the following:

1. 6 of the 9 mean results for male speech match the presented localization angle well. 4 of the 9 mean results were similarly matched for white noise, while 3 and 1 of the mean results matched for music and female speech sources, respectively.
2. All source signals were well localized with zero deviation at listening position 9, the closest to the presenting loudspeaker.
3. All source signals were similarly accurately localized at listener position 8. However the result for female speech showed a deviation of $\pm 13.6^\circ$.

The mean results show that male speech was localized with the highest degree of accuracy. As expected, localization blur is generally greater at the rear than for frontally positioned sources.

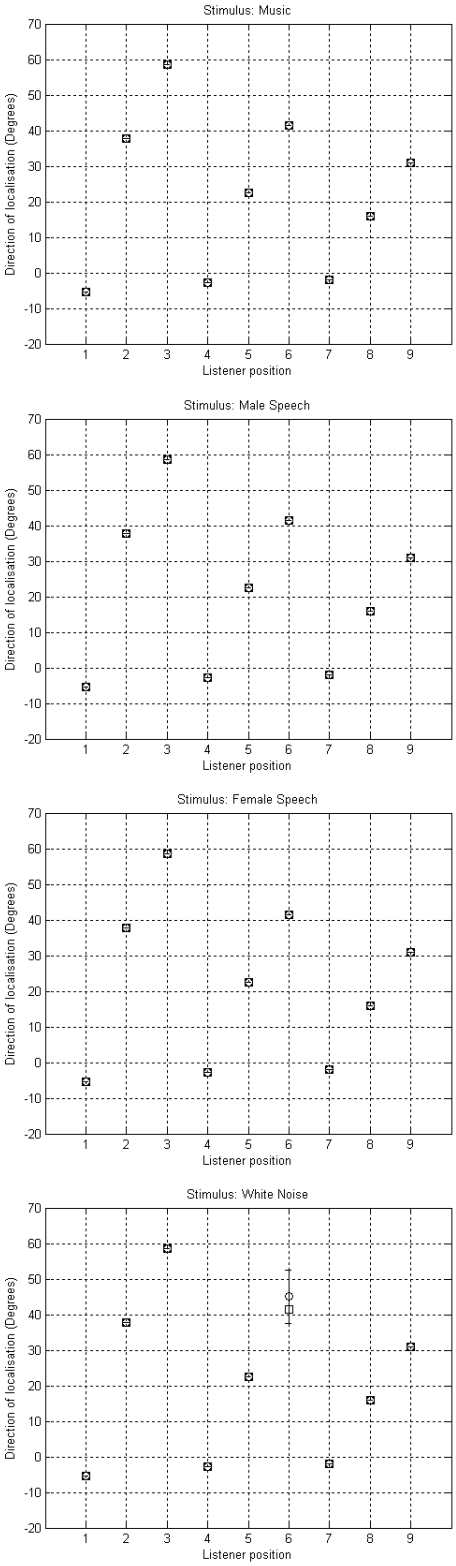


Figure 4: Subjective localization of source stimuli presented at loudspeaker 2 for all listener positions. $\bigcirc = \bar{\theta}$, $\square = \theta_T$, $\vdash\dashv = \pm\sigma_\theta$

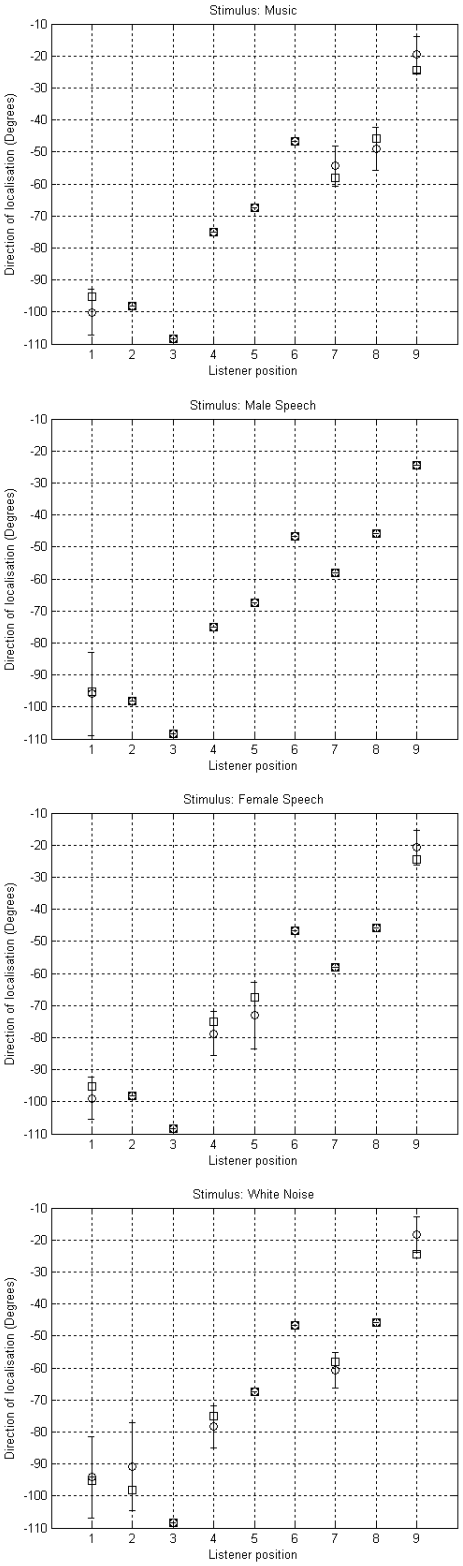


Figure 5: Subjective localization of source stimuli presented at loudspeaker 6 for all listener positions. $\bigcirc = \bar{\theta}$, $\square = \theta_T$, $\vdash\dashv = \pm\sigma_\theta$

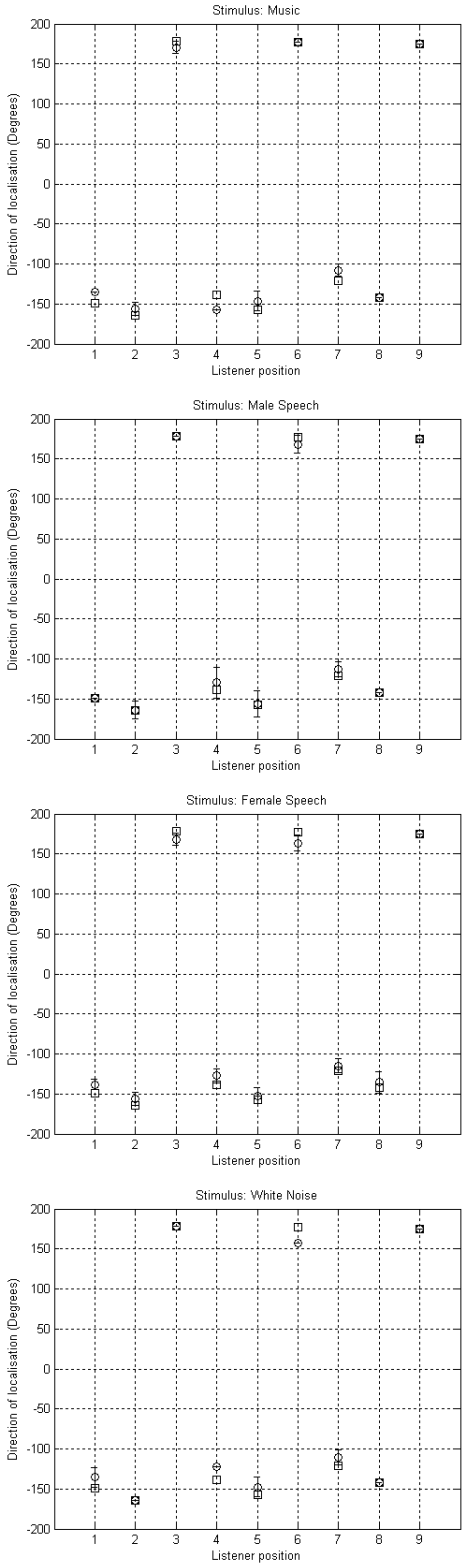


Figure 6: Subjective localization of source stimuli presented at loudspeaker 10 for all listener positions. ◯ = $\bar{\theta}$, □ = θ_T, ⊢⊣ = $\pm\sigma_\theta$

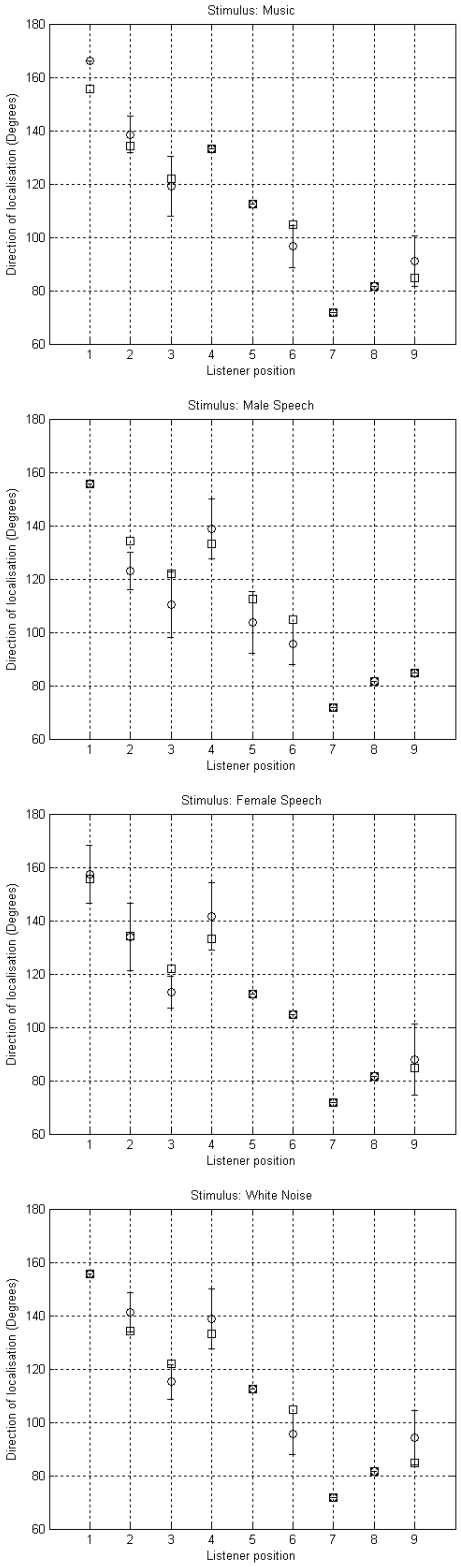


Figure 7: Subjective localization of source stimuli presented at loudspeaker 14 for all listener positions. ◯ = $\bar{\theta}$, □ = θ_T, ⊢⊣ = $\pm\sigma_\theta$

Figure 5 shows the results of a lateral presentation from speaker 6 (right, front). The results indicate the following:

1. The mean results for male speech all match the presented location angle with zero deviation, apart from listening position 1, which shows a deviation of $\pm 12.94^{\circ}$.
2. The highest number of results that matched the source position with zero deviation occurred for male speech (8), while 5 similar results occurred for the other three source signals.
3. Good localization was again achieved at the listening positions closest to the presenting loudspeaker (positions 3 & 6) for all source signals.

The mean results for this presentation indicate that all sources were reasonably well localized. Male speech again performed better than the other source signals.

Figure 7 shows the results of a second lateral presentation from speaker 14 (back, left). One can note the following:

1. 4 of the 9 mean results for each source signal match the presented localization angle.
2. All sources were well localized at the listening positions (7 & 8) closest to the presenting loudspeaker.
3. The results at other listening positions show wide deviations.

6. DISCUSSION

The above results indicate that monophonic sources can be reasonably well localized by a distributed audience under reverberant conditions. They also show that localization accuracy is greatest for frontal sources with a frontally-biased lateral source providing the next best results. The results for rear and rear-biased lateral sources were largely comparable.

The results for music, white noise and female speech were similar, while the best localization was achieved for male speech. Informal discussions after the experiment revealed that the general consensus among the test subjects was that white noise was the most difficult signal to localize. These impressions support the findings of other localization studies [1] which also indicate that localization accuracy is greater for speech than for broadband noise.

The subjects were tested using a forced-choice, speaker identification method which could explain the high degree of correlation between the mean results and presented angle. The range of deviation varies considerably for different listening and source positions which is unsurprising considering the non-ideal listening conditions. A number of studies [6, 27], have shown that localization accuracy decreases with increasing levels of reverberation. These findings were supported by our results which show wider angular deviations than reported in similar studies carried out under anechoic conditions [28].

In addition, one would expect the presence of early reflections and in particular, the lateral reflections typical of most concert halls, to similarly reduce localization accuracy. The test room contained a number of hard surfaces which presumably generated significant reflections. An analysis was therefore carried out on those test results where the mean significantly deviated from the presented angle. Listening positions 1, 4, 5 & 7 for a source at speaker 10 all display a negative angular bias. Likewise, the results at listening positions 6 and 9 for a source at speaker 14 all display a positive angular bias. These biases, combined with the close proximity of the loudspeakers to the walls seems to suggest the influence of lateral reflections on localization.

However, the difficulties in correlating angular biases such as these to specific reflections are well known and highly applicable here. In a previous study, Hartmann et al. proposed that when the azimuth of the reflection competes with the azimuth of a direct sound, subject's responses will be biased in the direction of the reflections [7]. He then went on to show that the effect of even a single reflection does not influence the perceived direction in this linear way. Data simulations and specific impulse response measurements could potentially reveal further information on the precise effect of early reflections on source localization in this particular case.

Although significant deviations were found, the results are nonetheless encouraging and seem to indicate a reasonable level of localization accuracy even under such non-ideal conditions. The angular deviation varied for different listening and source positions but never exceeded a maximum value of approximately 30°. An examination of the extreme situations, i. e. where the angle from a listening position to a pair of speakers is at a minimum, could help reveal the cause of these deviations. The results for frontal sources were highly accurate and so will not be considered here. The extreme condition for a source presented at speaker 6 occurs for listening position 9. The results indicate a deviation of approximately $\pm 6^{\circ}$ for this position with the mean result being biased by approximately 6° towards speaker number 4. The extreme condition for a source presented at speaker 14 occurs at listening position 1. The results show zero deviation for all sources except female speech ($\pm 10.93^{\circ}$) while the mean results match the presented angle for white noise and male speech, with the music source displaying a bias of 10.6° toward speaker 12. The extreme condition for a source presented at speaker 10 occurs at listening position 3. The results show zero deviation and a matching mean angle for white noise and male speech. The other sources display a deviation of approximately $\pm 8^{\circ}$ and a mean bias of approximately 10° toward speaker 8.

These results suggest that for most combinations of listening and source position, the localization blur is not sufficiently strong to cause a listener to localize a monophonic source away from the desired location when using an asymmetrical 8-speaker array. However, for extreme cases such as a front-corner listening position with a source positioned to the rear, accurate localization cannot be guaranteed. This problem appears to depend on the nature of the source signal.

7. CONCLUSION

The presented results for the given configuration and reverberant environment show that the localization of monophonic sources can be achieved well in a reverberant environment for a distributed audience. For the monophonic sources presented, it was found that, on average, the localization blur is not sufficient to cause localization away from the desired source direction. It was noted, however, that at extreme listener/source positions, the cues for accurate localization to the source angle may not be guaranteed with certain source stimuli. Furthermore, it was shown that the best stimulus for localization in a reverberant environment is male speech. Simulations and further empirical investigations to support the subjective tests of this research should also be undertaken.

These results also form a 'best case' scenario for any spatialization technique, since the presented environment pertains to a

real listening situation and not ideal anechoic conditions. Thus the best possible performance that spatialization schemes such as VBAP, DSS, Ambisonics and WFS can hope to achieve under similar conditions is that of the monophonic presentations shown. In light of this, the study undertaken provides a strong basis for the comparative studies of the performance of spatialization techniques in terms of localization accuracy and their technological relevance for music performance situations undertaken by the authors in [25, 26].

8. ACKNOWLEDGEMENTS

The authors wish to thank all the participants of the listening tests in particular the research postgraduates of the Department of Electronic and Electrical Engineering and the Music and Media Technology course, Trinity College Dublin. The assistance of Dr. Rozenn Dahyot is also greatly appreciated.

9. REFERENCES

[1] J. Blauert, *Spatial Hearing*, MIT Press Cambridge MA, 2003.

[2] J.W.S. Rayleigh, *Theory of Sound*, Dover, N.Y., 1945.

[3] E. A. MacPherson and J. C. Middlebrooks, "Listener weighting of cues for lateral angle: The Duplex Theory of sound localization revisited," *Journal of the Acoustical Society of America*, vol. 111, pp. 2219–2236, May 2002.

[4] S. G. Weinrich, "Horizontal plane localization ability and response time as a function of signal bandwidth," in *Audio Engineering Society Preprint 4007; AES Convention 98*, February 1995.

[5] T. T. Sandel, D. C. Teas, W. E. Feddersen, and L. A. Jeffress, "Localization of sound from single and paired sources," *Journal of the Acoustical Society of America*, vol. 27, pp. 842–852, July 1955.

[6] W. M. Hartmann, "Localization of sound in rooms ," *Journal of the Acoustical Society of America*, vol. 74, pp. 1380–1391, Nov. 1983.

[7] B. Rakerd and W. M. Hartmann, "Localization of sound in rooms, II: The effects of a single reflecting surface," *Journal of the Acoustical Society of America*, vol. 78, pp. 524–533, Aug. 1985.

[8] B. Rakerd and W. M. Hartmann, "Localization of sound in rooms, III: Onset and duration effects ," *Journal of the Acoustical Society of America*, vol. 80, pp. 1695–1706, Dec. 1986.

[9] A. Blumlein, "Improvements in and relating to sound transmission, sound recording, and sound reproducing systems," British Patent Specification 394325, 1931.

[10] V. Pulkki, "Virtual sound source positioning using vector base amplitude panning," *Journal of the Audio Engineering Society*, vol. 45, pp. 456–466, 1997.

[11] V. Pulkki, "Localization of amplitude-panned virtual sources i: Stereophonic panning," *Journal of the Audio Engineering Society*, vol. 49, pp. 739–751, 2001.

[12] D. G. Malham, "Experience with large area 3-D ambisonic sound systems," *Institute of Acoustics*, vol. 8, pp. 209–216, 1992.

[13] M. A. Gerzon, "Criteria for evaluating surround-sound systems," *Journal of the Audio Engineering Society*, vol. 25, pp. 400–408, 1977.

[14] J. M. Jot, V. Larcher, and J. M. Pernaux, "A comparative study of 3-D audio encoding and rendering techniques," in *16th International Conference of the Audio Engineering Society*, 1999, pp. 281–300.

[15] E. Benjamin, R. Lee, and A. J. Heller, "Localization in horizontal-only ambisonic systems," in *121st Convention of the Audio Engineering Society*, 2006.

[16] A. J. Berkhout, D. de Vries, and P. Vogel, "Acoustic control by wave field synthesis," *The Journal of the Acoustical Society of America*, vol. 93, no. 5, pp. 2764–2778, 1993.

[17] D. de Vries and P. Vogel, "Experience with a sound enhancement system based on wave front synthesis," in *Audio Engineering Society Preprint 3748; Convention 95*, 1993.

[18] T. Caulkins, E. Corteel, and O. Warusfel, "Analysis of certain challenges for the use of Wave Field Synthesis in concert based applications," in *Proceedings of the 7th International Conference on Digital Audio Effects (DAFX 04)*, October 2004, pp. 250–255.

[19] N. Sobol, "The DSP 610 - a computer controlled processor for a truly directional sound reinforcement system (the Delta Stereophony System)," in *Audio Engineering Society 6th International Conference*, 1988.

[20] W. Ahnert, "Complex simulation of soundfields by the Delta Stereophony System (DSS)," *Journal of the Audio Engineering Society*, vol. 35, pp. 643–652, 1987.

[21] P. Fels, "20 years Delta Stereophony System - high quality sound design," in *100th Convention of the Audio Engineering Society*, 1996.

[22] W. Hoeg, F. Steffen, and G. Steinke, "Delta Stereophony: A sound system with true direction and distance perception for large multipurpose halls," *Journal of the Audio Engineering Society*, vol. 31, pp. 500–511, 1983.

[23] P. Fels, "Multichannel and 'SOR' principles for conferencing and teleconferencing systems," in *110th Convention of the Audio Engineering Society*, 2001.

[24] W. Ahnert, "Problems of near-field sound reinforcement and of mobile sources in the operation of the Delta Stereophony System (DSS) and computer processing of the same," in *82nd Convention of the Audio Engineering Society*, 1987.

[25] E. Bates, G. Kearney, D. Furlong, and F. Boland, "Localization accuracy of advanced spatialisation techniques in medium-sized concert halls," in *153rd Meeting of the Acoustical Society of America*, June 2007.

[26] G. Kearney, E. Bates, D. Furlong, and F. Boland, "A comparative study of the performance of spatialisation techniques for a distributed audience in a concert hall environment," in *31st International Conference of the Audio Engineering Society*, June 2007.

[27] C Giguere and S. M. Abel, "Sound localization: Effects of reverberation time, speaker array, stimulus frequency, and stimulus rise/decay," *Journal of the Acoustical Society of America*, vol. 94, pp. 796–776, 1993.

[28] G. Theile and G. Plenge, "Localization of lateral phantom sources," *Journal of the Audio Engineering Society*, vol. 25, pp. 196–200, 1977.

SYNTHESIS OF A MACRO SOUND STRUCTURE WITHIN A SELF-ORGANIZING SYSTEM

Sinan BOKESOY

CICM, University of ParisVIII
Paris, France
sinan@sonic-disorder.com

ABSTRACT

This paper is focused on synthesizing macro-sound structures with certain ecological attributes to obtain perceptually interesting and compositionally useful results. The system, which delivers the sonic result is designed as a self organizing system. Certain principles of cybernetics are critically assessed in the paper in terms of interdependencies among system components, system dynamics and the system/environment coupling. It is aiming towards a self evolution of an ecological kind, applying an interactive exchange with its external conditions. The macro-organization of the sonic material is a result of interactions of events at a meso and micro level but also this exchange with its environment. The goal is to formulate some new principles and present its sketches here by arriving to a network of concepts suggesting new ideas in sound synthesis.

1. INTRODUCTION

Generally human made acoustical instruments produce compositionally meaningful sounds, which can be described by some interactions such as excitation and resonance. In such a structure, the excitation consists of a temporal energy input, a sort of some disturbance introduced to the system [1].When an acoustic resonant system is excited, it does attenuate some frequencies and emphasize certain ones. The resonance produces a pattern of energy dissipation. Such resonant systems reach a final stable state because energy is not generated within the system but it is received from an external source. And since the external source comes as an event trigger mechanism with a temporal energy supply having a transient behavior, the response of the system is finite. The interaction of such instruments with its environment happens at this excitation level. Although there might be involved chaotic dynamics on the micro level components of these systems, the macro level output should have a distinctive character as a response to a certain input. The output of the system should be predictable as it makes sense for the performer who would like to control it precisely with the input parameters of his performance. The interaction becomes in what ways the resonant system responds to its excitation system.

Our system which we are going to introduce as a sound synthesis instrument has a much more complicated structure than the one explained above. It will have a control mechanism which serves to the user interaction, but also a self organizing system to deliver dynamic behavior of eco-systemic kind. All this will be exploited within the principles of cybernetics.

2. SYSTEM STRUCTURE

Ecological systems have generally a hierarchy of multiple levels of organization on multiple time-rate scales in order to be ecologically valid [2]. Compositionally meaningful sound objects are subjected to a temporal change, spectral variation and envelope, which results as a pattern of change. This pattern becomes a structure establishing a form of the sonic identity perceivable by our ears such as we could be able classify the character of the sound source. In the case of ecologically valid sound objects, the patterns of change are expected to form higher-order percepts. In other words, within the hierarchy of multiple levels, the interaction of low level elements shows emergent properties at higher levels. This interaction occurs among all levels. Organization at one level influences the others [3].

In our system, we think the smallest element as a granular microsound event. Sounds constructed with the granular technique require high densities of short events to produce aurally convincing sound textures. Therefore, computer music composers have adopted algorithmic approach to handle granular synthesis with statistical controlled distributions combined with tendency masks, probabilistic functions and other methods while exploring the possibilities of controlling the granular streams [4] [5] [6]. In general, the lack of these applications for creating macro-temporal patterns was the employment of a mono-layered time structure, which was missing a hierarchical organization of multiple layers and complexity.

Figure 1: *An example sonic event distribution (represented by small dashes) inside consecutive time cells. The distribution in each cell is independent and stochastic. The arrow represents the possible information exchange between cells. But if this feature is not embedded in the structure, which would affect each others distribution process and there is no feedback from any output to input, no evolution is possible in such a structure.*

An alternative event distribution system operating on multiple time-scales within a self similar structure has been proposed [7] [8]. The event distribution mechanism on each level of Cosmos is using deterministic or stochastic functions for making decisions on each events onset time and duration parameters on that level (Figure 2). Each event opens a subspace with other events on a lower level in the structure. As we see on Figure 3, the mechanism of sound synthesis starts by injecting a sound element as the input on the micro level (forming the excitation system) and the bottom-up procedure constructs the meso-spaces and finally the macro-space. The micro level characteristics of a sound grain influence the meso and macro properties of the sound event. The output of the system is fed back again to the input, as the micro-event sample data. This recursive system uses its own output and creates a multi layered development in the sonic structure.

The particular transformations at the macro, meso and micro levels are user defined but they are fixed during the operation and non-adaptive. For example if any of the transformations features some 'destructive' processing, meaning that it leaves something out, the result of the feedback process is degradation.

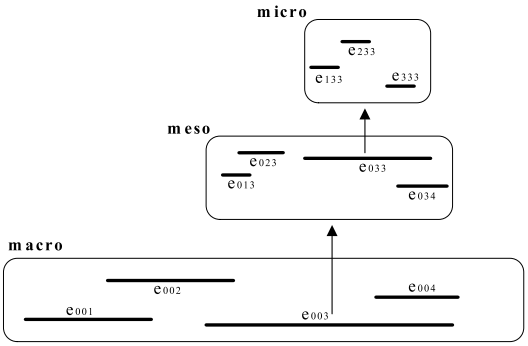


Figure 2: *The top-down sonic event distribution mechanism in of the Cosmos model. It is based on a self similar structure and interdependency between layers. The events are numbered sequentially, and some of them are overlapping while increasing the local density. Each macro event opens a meso-space and each meso-event opens a micro-space. The numbering process shows where each event is coming from.*

If the transformation is asynchronous granulation, the degradation results to a sonic powder; or if the output gains a noisy character then the feedback reinforces this behavior and the degradation of harmonic structure follows rapidly. But if there is any slight regular pattern in the function, it would be replicated at the output so it is not a drift towards white noise or distortion, and will have present a similar phenomena such as in multi-layered pulsar synthesis [6].

This behavior is the result of the positive feedback, which creates an attractor strong enough winning out all of the other features in the system. The feedback system soon points to its end-states.

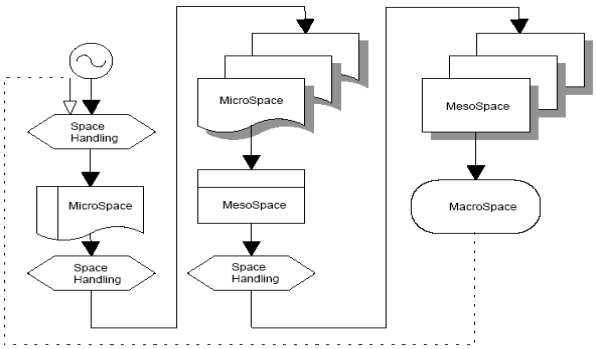


Figure 3: *This is the schematic overview of the audio routing and the event space organization inside the application. The audio enters top-left. The dotted line is the feedback audio line. The black arrows show the bottom-up organization of the sonic data. Each micro-space corresponds to a meso-event supplying the sonic data with the space-handling process and the meso events organize the meso-spaces corresponding to macro-events.*

What is being experiencing with degradation behavior here, is a lack of "structural coupling", a lack of "adaptation" to the input (although, for the moment, the input = the system output). In other words, the transformations, or their composition together at the macro level, are insensitive to the input/output, they just work the same way independent of what comes in. When such a system eliminates the destructive relationship to its input/output, it becomes an eco-system [9]. For establishing this, we should use the selected external conditions of the environment, which will be derived from the analysis of the output to maintain the perceptual attributes of sound such as; noisy/harmonic character, intensity, spectral distribution, rhythmical behavior, particle density.

2.1. Structure&Environment Coupling

Non-equilibrium systems are driven away from a stable position and exhibiting dynamic behavior. In many cases the transition period between states is significant (e.g. metamorphosis in insects takes time). This period, called a transient, is a non-equilibrium state (equilibrium here refers just a constant state, not only to the lowest energy state familiar from physics). It is the transients that are the actual behavior. What we have here is a closed stochastic system, so the steady state in our system is irrelevant. Complex systems of this sort never settle to a fixed status, maybe the only fixed state is the silence in our case. It is subject to constant perturbation (due to the input sound to the Cosmos, which drives bursts of transient behavior).

This instability with order is what we call 'Edge of Chaos', a system midway between stable and chaotic domains (self-organized criticality) 10]. It is characterized by a potential to develop structure over many different scales and is an often found feature of complex systems whose parts have some freedom to behave independently such as in Cosmos. For the 'edge of chaos' behavior we need some constraints – too many dynamics will die out, too few and absolute order will not be sustained.

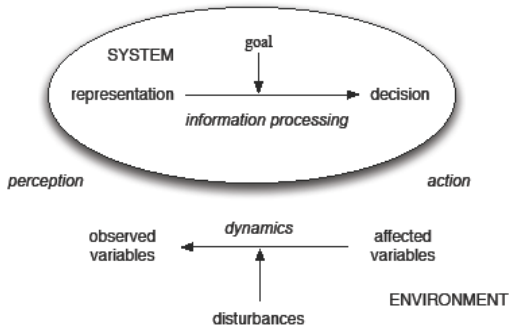


Figure 4: *A typical control mechanism in cybernetics. This model can be applied to many possible dynamic structures in different disciplines.*

The question is how to establish an evolutionary process of self-organization possible with this system. Because we enter the world of cybernetics, the system design should be considered within its principles [11]. Cybernetics was defined by Wiener as "the science of control&communication, in the animal and the machine", in a word, as the art of steermanship. Our system concept includes; complexity, system-environment boundry, process, state, hierarchy, feedback and network of coupled variables.

The aim would be first of all to create a dynamical system which will represent a structurally closed but organizationally open system [Figure 4]. It should be able to accept/create control commands for direct manipulation or self-organization within a coherent structure. Respecting this model, we can substitute Cosmos such as in [Figure 6]. What is fixed here is the structure of Cosmos, its integrity with its interconnected components. Cosmos should determine its state by the interactions with the environment and among its control system components by showing an adaptive behavior to the external parameters, which is essential to self-organization. This approach of cybernetics is taken universal and is valid for any sound-environment interaction, not specifically for the Cosmos model [9]. But the stochastic complex behavior of the Cosmos model introduces some interesting possibilities which will be discussed below.

2.1.1. The Observing Part

In our case, the system listens to its output, which becomes its 'environment interaction' (Figure 5). The output of the system itself is the perturbation introduced to the system against which it should react and organize its state. In an ideal system, to every class of perturbations there corresponds a class of adequate counteractions. This correspondence might be represented as a homomorphism from the set of perturbations in the environment to the set of compensations.

The observer part keeps tracking certain features of this environment and does an analysis on the sound for certain perceptual attributes, which are set at the design level. It becomes in the end a self-observing system, ultimately using information on its input/output, decides for the emergent behavior to take against the external conditions in order to re-organize itself.

Regarding the integral structure of Cosmos, if one implements some flexible behavior in any or all of the three levels, macro meso and micro, the representation of the selected external conditions will be evaluated and actions will be taken inside the new self organizing structure. The sound output of Cosmos itself can be interpreted with useful analysis methods to extract the sonic attributes [Figure 5]. The sound analysis can be further interpreted with statistical analysis tools to obtain the descriptive characteristics of the data sets with the arithmetic mean value $\overline{x}$, the standard-deviation σ, which is the most common measure of statistical dispersion, and the skew, which is a measure of the asymmetry of the distributed values for the incoming stream of observed values. The standard calculation method for these quantities is as in the equation (1).

$$\overline{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \quad , \quad \sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \overline{x})^2} \qquad (1)$$

$$skew = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{x_i - \overline{x}}{\sigma}\right]^3$$

There can be different integration times on these observed values corresponding to macro, meso and micro level time scales. The feature extraction process generates control rate data x_i, which is interpreted for the reaction of the system only possible after a certain delay time depending on these integration times.

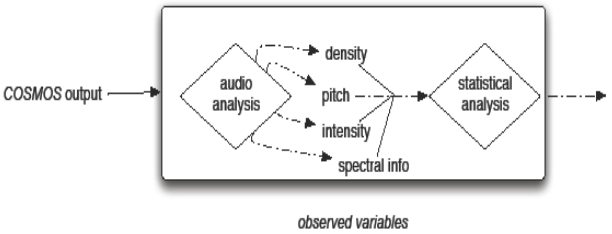


Figure 5: *The Cosmos output, which is the output of the system described in Fig3., is being analyzed and certain perceptually meaningful sonic parameters are extracted. Audio input comes in and control rate data comes out.*

- *intensity analysis* : At the very low level, the intensity detection is the basic observed parameter, where one can follow the envelope of the sound input to the system.

- *pitch analysis* : The pitch analysis can be interpreted in a range from deterministic output to noisy character. Also when the event distribution happens with a regular rhythmic order, then with fast distribution rates, the rate itself is being perceived as a pitch.

- *spectral processing* : Filtering methods emphasize a certain region of the spectra, therefore increasing the redundancy. Applied in a feedback loop, it is analogous to the Larsen tone effect, which is the true acoustical feedback. The use of several BandPass filters would allow following and matching certain regions in the spectrum of the incoming audio and serve as an analysis tool. The spectral centroid is also an elementary parameter, which can be observed for finding a correlating value for the spectral brightness of the incoming audio.

- *density* : The density here is the quantity of events in a certain time span. But when they are overlapping, the detection of the distinctive events becomes difficult although perceptually its existence is evident depending also to their pitch distribution. When the pitch distribution is wider, the perception of discrete events becomes easier. The more density of the overlapping events, the more becomes the perceived intensity. Therefore density and intensity are correlated.

For the implementation on Max/MSP, the 'analyze~[1]' object can be used to determine the *pitch, loudness, noisiness* and *brightness*. The loudness is already an average value, because the STFT is being used with an overlapping window function which does extract the spectral contour along the bins by averaging their intensity along the window size. The object 'lp.stacey[2]' on Max/MSP can be used to report the statistical analysis for the *mean value, standard deviation* and *skew* quantity; but also one can implement these functions easily with JavaScript using already existing source codes.

Furthermore one can apply low-pass filter for smoothing the noisy data. The Savitzky-Golay smoothing filter [12] is an example and can be used before the statistical description stage of the data. A second order polynomial has been used to filter the noisy part from the data, which is easily implemented on MaxMSP.

We are not going to dive here into details and problematic in extracting the features like the precise pitch, amplitude and density values on the incoming data in terms of reliability. It is the reason, why we have suggested having a descriptive analysis of the data with the statistical functions. It is more efficient in our case to use the statistically significant output of the analysis and make use of this on the decision part. The artifacts of the analysis represent partly the incompleteness regarding the representation of the external conditions. But also according to the cybernetics principle of 'incomplete knowledge' [13], the model embodied in a control system is necessarily incomplete, the system cannot represent itself completely, and hence cannot have complete knowledge of how its own actions may feed back in to the perturbations.

2.1.2. The Decision Part

Adaptive behavior can be at any time scale, and can follow the perceptual attributes and react by changing the system variables depending on these conditions. In order to adequately compensate perturbations, a control system must "know" which action to select from the variety of available actions.

1 *analyze~* object developed by Tristan Jehan

2 *The Litter Pack* objects developed by Peter Castine

This is the law of 'Requisite Knowledge of Cybernetics' [13]. Without the knowledge the system would act blindly to the external conditions. The analysis results should be represented in the Cosmos model with the relevant system variables such as when the system tries to fulfill some goals, the Cosmos model should contain the representation of selected decisions in reaction to the external conditions. Likewise this adaptive behavior leads and forces the system state to change itself. [2]

The system variables in the Cosmos model are the *onset and duration distribution functions* (a range from deterministic to various stochastic functions), *density distribution functions*, and *stochastic modulation generators* which do affect pitch, intensity, filter parameters on macro, meso and micro levels independently. The overall goal of these functions is to achieve control on each event space and perform the process of change on the appropriate operation level. This organized spatial distribution of events and modulation functions are reminiscent to *morphogenesis* in developmental biology, where it is the study about understanding the processes that control the organized distribution of cells during the development of an organism. This change is controlled by the genetic program and can be modified by the environmental factors. The decision part in our system, should deliver this genetic code in controlling the 'organic' function of Cosmos, which develops the morphological aspects of sound.

The challenge of the user is how to describe the organic character and translate it in the decision mechanism with the available parameters. This involves the classification of the macro-sound structures. The problematic in this classification effort is also the definition of the spectral sound morphology, which is a process of change, a transition between states in the timbre space [14]. Everything is a matter of degree. Within this scheme, for the future implementation, we suggest to use fuzzy logic operations in its linguistic form to allow partial membership in a set of macro sound characteristics. Therefore, this fuzzy inference step takes control in the decision module, where the statistical analysis coming from the observer module is the input parameter and the macro sound representation is the output. One can use the fuzzy logic control kit [15] for the implementation on MaxMSP.

2.1.3. Preliminary ideas on the implementation

Implementing and regulating the system behavior in Cosmos is complex since the interdependencies among these system variables are subject to create unexpected emergent behavior and they indirectly implement the system dynamics. There are two possible situations of emergent behavior here;

- Within Cosmos; unpredictable emergent behavior because of the stochastic self similar structure. It does emerge and maintain itself at the 'Edge of Chaos'.
- The action which the system takes for organizing itself in the direction of the decision mechanism. It does produce order bottom-up.

The decision mechanism, which is designed as an external application, will decide for the manipulation and application of these i/o functions according to the desired goal [Figure 6]. The intervening user could assign the settings at the initialization point.

With regard to any parameter observed, the decision part can be a conformer or a regulator at the most basic level. The parameter which is compared by the decision module, is going to be observed inside certain boundries for an statistical inspection of the value. This distribution range and the boundries will be set also by the user.

parameter value + distribution range ⟶ boundries of parameter space

- Regulation tries to maintain the parameter in the system at a constant level, regardless what is happening in the external conditions.
- As a conformer, it allows the environment to determine the parameter, therefore applying a positive feedback in adapting itself to the external conditions.

The output of Cosmos represents the emergent behavior including the unpredictable elements in the bottom-up development. It is also the perturbation which enters to the system and causes the transient action. Also represented in Figure 6, the chain of actions will be like below in the system.

- The observer does the analysis;
- The decision module compares the results with its internal conditions and decides what action it should take against the incoming external conditions;
- The reactions are mapped to the Cosmos parameter space in order to represent them inside the system;
- The internal conditions of the decision module are set by the user, which reflect a compositional function utilized by the user;

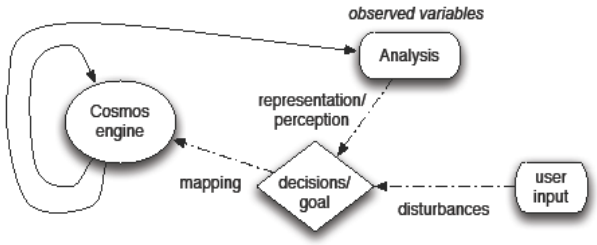


Figure 6: *The Cosmos engine, which is the system described in Fig3., is being observed and represented inside the control system. The user interacts with the decision module, which establishes control by mapping its decisions to Cosmos parameter space.*

The delay between the decision point and the observation beginning is merely based on the observation process length depending on the macro, meso and micro level processes in Cosmos. For instance, if we would like to analyze the process of change on the macro-space in Cosmos, the data will arrive after the macro-space duration, and the decisions will be taken and applied at the start of the next macro-space. The more the delay time, the more difficult establishing the precise control. At some point, transients may not have enough time to settle down because of the late reaction of the decision mechanism and the system will exhibit chaotic behavior.

3. DYNAMIC MODES OF OPERATION

What we see in first case is a dynamic complexity, where the structure of a system may be simple, but the behavior unpredictable. It is a property of its behavior. Some authors define that complex systems may be still divided up to complex adaptive systems and complex deterministic systems. In a deterministic system it is always possible to predict the final state, if the initial state is known. In a complex adaptive system it is not possible to know the initial state in such a detailed level, that the final state could be predicted. This is due to the non-linearity and the loops of positive feedback.

It is expected that the implemented control system establishes an attractor, where the behavior of this complex stochastic system with unpredictable patterns would approach to a configuration of the phase state characteristic to the control system attractor. The goal-directedness of this system is suppression of these unpredictable patterns and deviations from the basin of the attractor. Musically a steady state tone or silence can be regarded as a stable state and noise as a chaotic state for the system. What is interesting here is the other equilibrium states for the observing mechanism which are at the 'edge of chaos'. So in going from any state to one of the equilibriums (the goal presented by the decision mechanism), the system is going from a larger number of states to a smaller. In this way, it is performing a selection and this reduction in the number of reachable states signifies that the variety, and hence the statistical entropy of the system diminishes. This is called again the process of self-organization.

In general, a complex system will have separate dynamic modes of operation. We are considering our system as a dissipative system, which take energy input (its own output as the audio material feedback and the control data from the decision mechanism) to maintain its homeostatic position. It is the flow of matter and energy through the system that allows the system to self-organize, and the exchange entropy with the environment.

Homeostasis is the property of an open system, the self-maintaining nature of systems from the cell to the whole organism [16]. Reactive homeostasis in biological systems is an immediate response to a homeostatic challenge such as predation. This predation is depending on the structure implemented in the event distribution and modulation mechanism of Cosmos; the ability of its morphogenesis. Especially in sound synthesis there can be many low level sound operations which deliver non-linear, non-reversible processes such as the frequency modulation technique, where there is a non-linear relationship between the input spectra and the output spectra. *Homeostatis* is a feedback phenomenon which cannot exist without reaction. In our case with Cosmos, this reactive compensation is reestablished by finding the desired internal state according to the decision part of the system.

Will the output of the system respect exactly the desired control, or will it be asymptotically close to its destination? (a question regarding the evolution theory) How do we define the attractor from a compositional point of view?

The environment which is introduced by the user on the decision mechanism is a pre-defined artificial space with its statistically significant perceptual attributes. With these parameters one specifies the attractors characteristic for certain musical/sonic features.

If the combination of these control parameters offers the variety of an environment which becomes the desired timbre space, the perturbations in the input would lead the system to re-organize itself approaching the attractors within this timbre space.

4. SOME COMPOSITIONAL ASPECTS

Stockhausen[3] states that any separation between acoustics and music is no longer meaningful in this era of computer aided sound design. The line between musically interesting synthetic sounds and digital sound effects can be very thin. Therefore the use of such a system is interesting in both ways. There are intriguing compositional aspects necessary to mention here and we claim that the philosophical structure is compatible with the ideas of musical sound&form having progressed since the beginning of 20th century. John Cage[4] has regarded the form, the structure (the divisibility of a whole into parts) as the expressive content, the morphology of the continuity. The simultaneous morphologies in different dimensions, which focus on particular perceptual attributes of sound, lead to the existence of form. Luigi Russolo[5] has assessed that the musical art is searching the most dissonant combination of sound, the most strange and strident, namely a musical noise. Algorithmic composition has dealt in the beginning with the generation or transformation of notes and phrases. In this case, the composition of larger temporal forms is a process of both composing the phases and also organizing them into larger structure (Iannis Xenakis[6]). Within the ability to reach the microstructure of sound, the process above has become the composition of sound and then composing with the sound [17]. At this point the note event is no longer assigned to specific sound sources as we call them performed instruments, and particularly the process of composing the sound itself has gained the ability to deliver the formal structure and complexity by accessing the morphologies in different dimensions of the structure.

The system which we have presented here is, aiming towards shaping the timbre space in that direction too. The indeterminacy is built in due to the unpredictable emergent behavior. We define the restrictions and constraints at the decision module, which demands the system to re-organize itself and shape the complex sonic output according to the compositional needs. The constraints let the sound object still evolve organically within these dynamics, so it would be a compositional idea to define the balance point, the distance from the attractor defined by the environment; the requisite variety versus the requisite constraint. The question would be what is the musical perceptual meaning of this trade point?

The "fitness" of states in the system is determined by how closely they match the formal needs of the particular dimensions set in the decision module. In the sense of the direction of evolution, what would be the average fitness with compositional means? Can we classify this with existing terms like harmonic, non-harmonic, order-disorder, and with general terms describing the macro-sound object? These questions are pushing this research forward and suggesting the planning for future work.

3 in *Perspectives of New Music* 1(1), 39-48, 1967

4 in his essay on "I*ndeterminacy*" 1958

5 *L'Art des Bruits*, Manifeste Futuriste 1913

6 beginning with his *Achorripsis* 1956 and *ST* series compositions 1956-62

5. ACKNOWLEDGEMENTS

I'd like to thank Agostino Di Scipio for his inspiring work and for his support in leading me with his suggestions about this research.

6. REFERENCES

[1] D. Keller, touch'n'go: Ecological Models in Composition. Master of Fine Arts Thesis, Simon Fraser University. Burnaby, BC, 1999.

[2] D. Keller., B. Truax., Ecologically-based granular synthesis in *Proceedings of the International Computer Music Conference*. Ann Arbor, MI: ICMA. 1998.

[3] Bregman A.S., *Auditory Scene Analysis: The Perceptual Organization of Sound*.Cambridge, MA: MIT Press, 1991.

[4] I. Xenakis, Formalized Music- Revised Edition. New York: Pendragon Press, 1992.

[5] B. Truax., "Real-time granular synthesis with a digital signal processing computer" in: Computer Music Journal 12(2), pp. 14-26.,1987.

[6] C. Roads., "Sound Composition with Pulsars" in : *Journal of the AES*, vol. 49, no. 3, pp. 134-147, March 2001

[7] S. Bokesoy., "The Cosmos model, an event generation system for synthesizing sonic structures", in *Proc. of International Computer Music Conference (ICMC'05)*, Barcelona, Spain, pp. 259-262, 2005.

[8] S. Bokesoy., "Feedback Implementation within a Complex Event Generation System for Synthesizing Sonic Structures" in *Proc. of Digital Audio Effects (DAFx'06)*, Montreal, Canada, pp. 199-203., 2006.

[9] A.. Di Scipio., "Sound is the Interface" in *Proceedings of the Colloquium on Musical Informatics*, Firenze, Italy, 2003

[10] C. Lucas, *Perturbation and Transients-The Edge of Chaos*, available at http://www.calresco.org/perturb.htm, accessed February, 2007.

[11] W.R. Ashby., *Introduction to Cybernetics*, Methuen, London. 1956-1999 (electronically republished at http://pcp.vub.ac.be/books/IntroCyb.pdf)

[12] W. Press and S. Teukolsky., *Numerical Recipes in C*, Second Edition, Cambridge,University Press, pp. 650-655, 1997.

[13] F. Heylingen., *Principles of Systems and Cybernetics: and evolutionary perspective*, in Cybernetics and Systems 92, R. Trappl (ed.), (World Science, Singapore), pp. 3-10, 1992.

[14] D. Smalley., *Spectro-Morphology and Structure Processes*, in: S. Emmerson (ed.) The Language of Electroacoustic Music Basingstoke: Macmillan, pp. 61-93, 1986.

[15] R. Cadiz and Gray S. Kendall, "Fuzzy Logic Control Tool Kit: Real-Time Fuzzy Control for Max/MSP and Pd" in *Proc. of International Computer Music Conference*, New Orleans, 2006.

[16] N. Wiener., Cybernetics (Control and Communication in the Animal and the Machine), MIT Press, Cambridge MA, 1948,1961.

[17] A. Di Scipio., "Formal Processes of Timbre Composition Challenging the Dualistic Paradigm of Computer Music" in *Proceedings of the International Computer Music Conference*. Aarhus, ICMA. 1994

CHARACTERISTICS OF BROKEN-LINE APPROXIMATION AND ITS USE IN DISTORTION AUDIO EFFECTS

Jiri Schimmel
Brno University of Technology, FEEC
Brno, Czech Republic
schimmel@feec.vutbr.cz

Jiri Misurec
Brno University of Technology, FEEC
Brno, Czech Republic
misurec@feec.vutbr.cz

ABSTRACT

This paper deals with an analytic solution of spectrum changes in scalar non-linear discrete systems without memory, whose transfer characteristics can be approximated via broken-line function. The paper also deals with relations between the harmonics ratio and the approximation parameters. Furthermore, the dependence of the harmonics ratio on the amplitude of a harmonic input signal is presented for the most common characteristics that are approximated via broken-line function. These characteristics are judged from the dissonance point of view.

1. INTRODUCTION

We have to deal with aliasing when a digital signal is being processed by a non-linear system. The aliasing is caused by bandwidth extension when the highest frequency component exceeds half the sampling frequency. To prevent that, we can either upsample the input signal or approximate the transfer function of the system via the finite sum of terms of Taylor's series and use nonlinear processing by band-limiting input signal range as published in [1].

That is why non-linear systems are used with such a type of approximation whose response to an input signal with limited bandwidth has a limited bandwidth as well (e.g. polynomial approximation) or with such a type of approximation which ensures that harmonics of a higher order than n are masked by harmonics of a lower order than n or with non-stationary spectrum components (e.g. exponential approximation [2]).

The polynomial and exponential approximations have the advantage that approximation parameters can be evaluated by solving a linear equation system according to the required ratio of harmonics [2]. On the contrary, the ratio cannot be set-up independently for each harmonic when broken-line approximation is used. Furthermore, the response of such a system to a limited-bandwidth input signal has not a limited bandwidth (see below). However, the computing-power demands of these systems are low. An analytic solution to the computation of amplitude of higher harmonics will be presented below as well as the common spectrum types of output signal, which can be produced by a non-linear system with broken-line approximation as a response to the harmonic input signal.

2. BROKEN-LINE APPROXIMATION

Broken-line approximation of a non-linear transfer function $\Psi(\ )$ is defined using R linear sections, for which the following equation holds for $i = 1, 2, ..., R$

$$y[n] = S_i\left(x[n] - x_{\mathrm{P}i}\right) \quad \text{for } x_{i-1} \le x[n] < x_i\,, \tag{1}$$

where S_i are the slopes of straight lines in the area $x_{i-1} \le x[n] < x_i$, $x_{\mathrm{P}i}$ are the points in which given line cuts the x axis, and x_i are the lower limits of a particular section of the function (see Figure 1). In the discrete domain, the spectrum changes can be evaluated in such systems by computing the approximation of coefficients of the discrete Fourier transform [3].

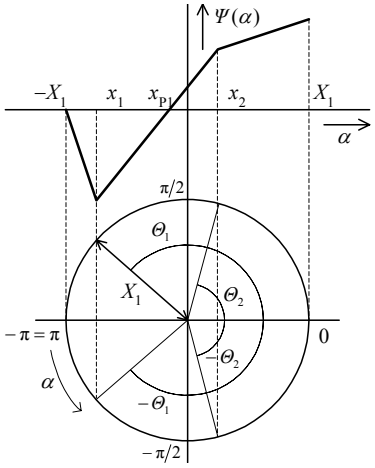


Figure 1: *Drawing up the output signal equation of non-linear system with transfer characteristics approximated by broken-line function using limit angles.*

The even-function attribute of the Fourier series [3] can be utilized in the case of spectral component analysis of the output signal of a non-linear system with cosine input signal (which is an even function). It can be seen from Figure 1 that the period of input signal $x(\alpha)$ is 2π and the function $\Psi(x)$ is identical for $x(\alpha)$ and for $x(-\alpha)$. The following equation holds for the approximation of the discrete Fourier transform coefficients of output signal

$$\begin{aligned} Y(\alpha) &= \frac{1}{\pi}\int_{-\pi}^{\pi} y(\alpha)\cos k\alpha \,\mathrm{d}\alpha = \\ &= \frac{1}{\pi}\int_{-\pi}^{0} y(\alpha)\cos k\alpha \,\mathrm{d}\alpha + \frac{1}{\pi}\int_{0}^{\pi} y(\alpha)\cos k\alpha \,\mathrm{d}\alpha \end{aligned}\,. \tag{2}$$

If we substitute equation (1) to equation (2) we obtain the following equation

$$Y(\alpha) = \frac{X_1}{\pi}\sum_{i=0}^{R-1} S_i\left(\int_{-\Theta_i}^{-\Theta_{i+1}} \mathrm{P}_{k,i}(\alpha)\mathrm{d}\alpha + \int_{\Theta_{i+1}}^{\Theta_i} \mathrm{P}_{k,i}(\alpha)\mathrm{d}\alpha\right), \tag{3}$$

where $\Theta_0 = \pi$, $\Theta_R = 0$, and

$$\mathrm{P}_{k,i}(\alpha) = \cos\alpha\cos k\alpha - \frac{x_{\mathrm{P}i}}{X_1}\cos k\alpha \,. \tag{4}$$

Using several goniometrical identities we obtain the following equation for the amplitude of k-th harmonics for $k > 1$ (see [2] for details)

$$Y_k(\alpha) = \frac{2X_1}{\pi}\sum_{i=1}^{R-1} \frac{S_i - S_{i-1}}{k^2-1}\left(k\sin k\Theta_i\cos\Theta_i - \sin\Theta_i\cos k\Theta_i\right) - \frac{S_i x_{\mathrm{P}i} - S_{i-1}x_{\mathrm{P}i-1}}{kX_1}\sin k\Theta_i, \tag{5}$$

where $\alpha = 2\pi n/N$, N is the length of the processed signal, X_1 is the amplitude of the harmonic input signal, and $\cos\Theta_i = x_i/X_1$. The following equations hold for the amplitude of the first harmonic and the dc component [2]

$$Y_1(\alpha) = \frac{X_1}{\pi}\sum_{i=1}^{R-1} (S_i - S_{i-1})(\Theta_i + \cos\Theta_i\sin\Theta_i) - \frac{2}{X_1}(S_i x_{\mathrm{P}i} - S_{i-1}x_{\mathrm{P}i-1})\sin\Theta_i \tag{6}$$

$$\frac{Y_0(\alpha)}{2} = \frac{X_1}{\pi}\sum_{i=1}^{R-1}(S_i - S_{i-1})\sin\Theta_i - (S_i x_{\mathrm{P}i} - S_{i-1}x_{\mathrm{P}i-1})\frac{\Theta_i}{X_1}. \tag{7}$$

2.1. Characteristics and Parameters of Approximation

It can be seen from equation (5) that the output signal of a non-linear system with transfer characteristics approximated by broken-line function has not a limited bandwidth if $R > 1$. Equation (5) is a sum of goniometric functions, and a period ξ of spectral component amplitude repetition can be found for a finite number of limit angles Θ_i. However, their amplitudes decrease very slowly. The highest harmonic of the output signal spectrum that is not masked by lower harmonics can be found using the psycho-acoustical model. The upsampling ratio can be chosen according to the order of this harmonic.

Furthermore, equation (5) shows that the amplitudes of harmonics depend on the difference of slopes $S_i - S_{i-1}$ of adjacent linear sections, rather than on the difference $S_i x_{\mathrm{P}i} - S_{i-1}x_{\mathrm{P}i-1}$. So the amplitudes of higher harmonics increase with the difference of right and left limits at the points of function discreteness. The following equations hold for $i = 0,1,\ldots, R-1$ if the approximation function is continuous

$$S_{i-1}(x_i - x_{\mathrm{P}i-1}) = S_i(x_i - x_{\mathrm{P}i}), \tag{8}$$

i.e. the S_i and $x_{\mathrm{P}i}$ parameters are linearly dependent according to the equation

$$x_{\mathrm{P}i} = x_i + \frac{S_{i-1}}{S_i}(x_{\mathrm{P}i-1} - x_i). \tag{9}$$

By equation (5) one could say that the amplitudes of output signal harmonics are linearly dependent. However, the substitutions $x_0 = -X_1$ a $x_R = X_1$ were used when equation (5) was derived (see Figure 1). So the input signal amplitude changes influence all limit angles Θ_i. The input signal will not span the i-th section of the characteristic if $|\cos\Theta_i| > 1$. The step-change of spectrum character of the output signal can be seen from the joint amplitude-frequency analysis in Figure 2, when the input signal amplitude exceeds the limit level $\cos\Theta_i$. The properties of the odd and the even transfer characteristic function can be also demonstrated using equation (5). With the harmonic input signal, only odd harmonics will be in the output signal when $\Psi(\alpha) = -\Psi(-\alpha)$, and only even harmonics will be in the output signal when $\Psi(\alpha) = \Psi(-\alpha)$ (see [2] for proof).

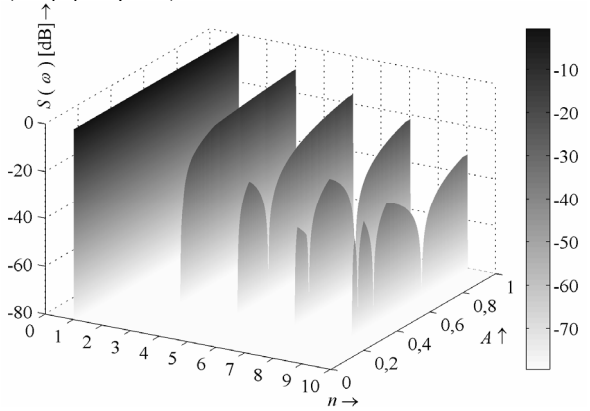


Figure 2: *Joint amplitude-frequency analysis of output signal of symmetrical limiter when* $\cos\Theta = 0.3$.

In the common case the broken-line approximation of transfer characteristic with R linear sections has $3R-1$ parameters. The number of parameters decreases when the S_i and $x_{\mathrm{P}i}$ parameters are linearly dependent according to equation (8). The Following equations hold for the $x_{\mathrm{P}0}$ a S_{R-1} parameters if the output signal range is $\langle y_{\min}; y_{\max}\rangle$

$$\begin{aligned} x_{\mathrm{P}0} &= -X_1 - y_{\min}/S_0 \\ S_{R-1} &= y_{\max}/(X_1 - x_{\mathrm{P}R-1}) \end{aligned}. \tag{10}$$

The total number of parameters of broken-line continuous function with R sections is $2R-2$.

2.2. Characteristics of Typical Broken-Line Approximations

Figure 3 shows the transfer characteristics of a simple non-linear system.

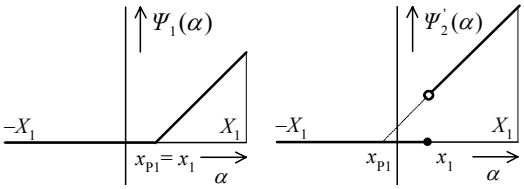


Figure 3: *Transfer characteristics of simple non-linear system.*

Equations for computing the amplitudes of output signal harmonics for such a type of system can be found in various publications dealing with analogue non-linear systems (e.g. in [4]). We can obtain the same equations for $\Psi_1(\alpha)$ from equation (5)

$$\frac{Y_k(\alpha)}{Y_1(\alpha)} = \frac{2}{k(k^2-1)}\frac{\sin k\Theta\cos\Theta - k\sin\Theta\cos k\Theta}{\Theta - \sin\Theta\cos\Theta}. \tag{11}$$

The following equation holds for a modified function $\Psi_1'(\alpha)$

$$Y_k'(\alpha) = Y_k(\alpha) + \frac{2S\delta}{k\pi}\sin k\Theta \text{ for } k = 1,2,\ldots, \tag{12}$$

where $\delta = x_1 - x_{\mathrm{P}1}$. Figure 4 shows the joint amplitude-frequency analysis of output signal of a non-linear system with this type of transfer characteristic. The level of the output signal is zero for

input signal amplitudes below x_1. The amplitudes of higher harmonics are high if level x_1 is slightly exceeded, and decrease with increasing harmonic input signal amplitude.

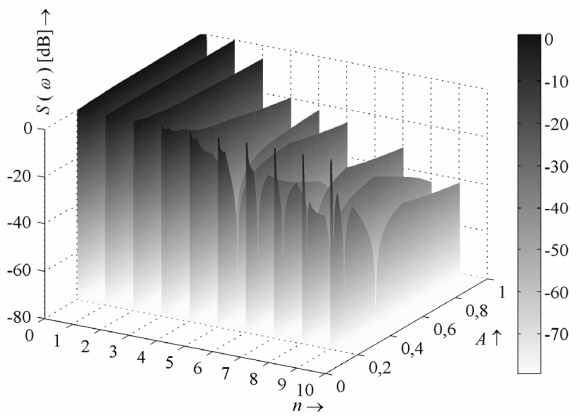


Figure 4: *Joint amplitude-frequency analysis of output signal of non-linear system from Figure 3.*

Figure 5 shows an interesting output signal spectrum that we obtain for $x_1 = 0$. In this case, the spectrum of the output signal consists of the first and the even harmonics only and their amplitude ratio does not depend on the amplitude of the harmonic input signal.

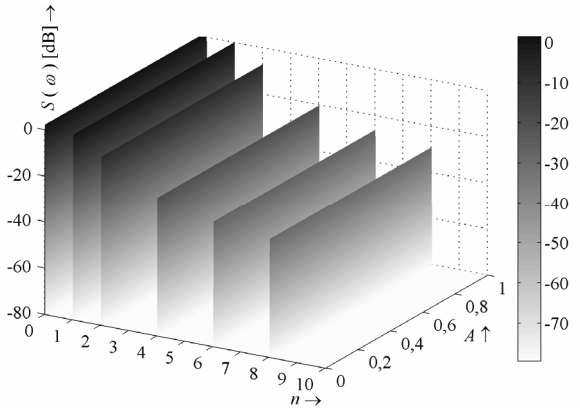


Figure 5: *Joint amplitude-frequency analysis of output signal of non-linear system from Figure 3 with $x_1 = x_{P1} = 0$.*

Figure 6 shows the transfer characteristics of a system with soft and hard thresholds.

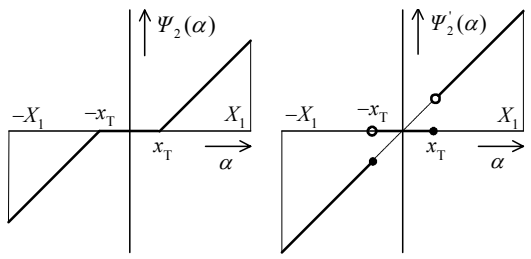


Figure 6: *Transfer characteristics of system with soft and hard threshold.*

We can derive equations for this system from equation (5), which are similar to equations (11) and (12). However, the even harmonics have zero amplitude and the amplitude of the odd harmonics is doubled.

On the contrary, the amplitudes of the odd harmonics are zero and the amplitudes of the even harmonics are doubled in comparison with equations (11) and (12) when the transfer characteristics are abs($\Psi_2(\alpha)$) and abs($\Psi_2'(\alpha)$) (see [2] for proof).

Typical transfer characteristics of non-linear system commonly used for distortion audio effects are in Figure 7. We can derive relatively complicated equations for amplitudes of the harmonics of output signal spectrum of a non-symmetrical limiter ($x_{T1} \neq -x_{T2}$) from equation (5) (see [2] for details).

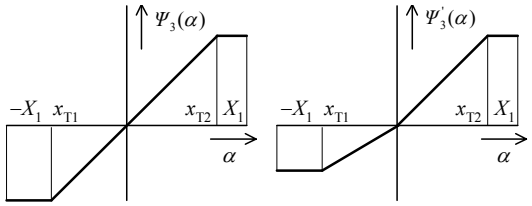


Figure 7: *Transfer characteristics of non-symmetrical limiter and limiter with non-linearity around operating point.*

Figure 8 shows the joint amplitude-frequency analysis of the output signal of non-symmetrical limiter.

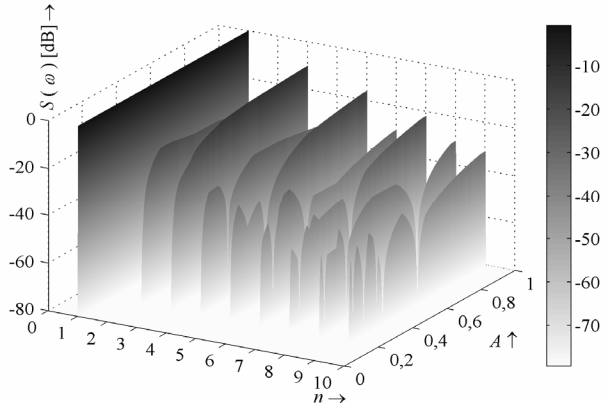


Figure 8: *Joint amplitude-frequency analysis of output signal of non-symmetrical limiter.*

Simpler equations can be derived for the symmetrical limiter with transfer function $\Psi_3(\alpha)$ when $x_{T1} = -x_{T2}$

$$Y_k(\alpha) = \frac{2S_1X_1}{\pi(k^2-1)}\left(1+(-1)^{k+1}\right)\left(k\sin k\Theta_2\cos\Theta_2 - \sin\Theta_2\cos k\Theta_2\right) \quad (13)$$

$$Y_1(\alpha) = \frac{2S_1X_1}{\pi}\left(\Theta_2 + \sin\Theta_2\cos\Theta_2\right) \quad \frac{Y_0(\alpha)}{2} = 0. \quad (14)$$

One can see from equations (13) that the amplitudes of the even harmonics are zero (see Figure 2). On the contrary, the amplitudes of the odd harmonics of symmetrical limiter with transfer function abs($\Psi_3(\alpha)$) when $x_{T1} = -x_{T2}$ are zero. It can be seen from the following equations derived for such a type of system from equation (5)

$$Y_k(\alpha) = \frac{2S_1X_1}{\pi(k^2-1)}\left(1+(-1)^{k}\right)\left(k\sin k\Theta_2\cos\Theta_2 - \sin\Theta_2\cos k\Theta_2\right) \quad (15)$$

$$Y_1(\alpha)=0 \quad \frac{Y_0(\alpha)}{2}=\frac{2S_1X_1}{\pi}\left(\Theta_2\cos\Theta_2+\sin\Theta_2\right). \tag{16}$$

The transfer characteristics of all systems mentioned above have a linear section around the operating point. That is why only the first harmonic (or no signal) is present at the system output until the amplitude of the input harmonic signal exceeds the first limit point. Figure 9 shows the joint amplitude-frequency analysis of output signal of limiter with transfer characteristics $\Psi'_3(\alpha)$ from Figure 7, which has non-linearity around the operating point.

It can be seen that the output signal spectrum for the amplitudes of harmonic input signal below the first limit point is similar to the output signal spectrum of the non-linear system from Figure 3 with $x_1 = x_{P1} = 0$. Higher harmonics ratio depends on the difference of slopes $S_i - S_{i-1}$ as mentioned in section 2.1.

2.3. Distortion DAFx Using Broken-Line Approximation

The scalar non-linear discrete system without memory, whose transfer characteristic can be approximated via broken-line function, can be used in any nonlinear audio processor.

The question is which type of approximation should be used for the distortion audio effect. Several typical approximations used in these effects are described in [5] which start from analogue prototypes. However, we can design a non-linear system that generates higher harmonics according to our requirements using equation (5) and equations derived from it. We assume that such a type of spectrum enhancement of the output signal is required that is not perceived unpleasantly. Furthermore, we assume that such an upsampling ratio is used that the aliasing spectrum components are masked by the harmonic components.

There are several criteria for the valuation of non-linear distortion of a system, e.g. simple valuation using weighted harmonic distortion. The valuation using the dissonance ratio is another type of valuation. It is most frequently determined as the multiplication of numerator and denominator of a fraction that determines the interval between two pure tones. A 2D histogram can be obtained if this valuation is applied to intervals between the harmonic spectrum components. The dissonance ratio increases with the number of the harmonic, it is lower with the even harmonics, and it is highest with the seventh harmonic (see [2] for details).

According to this valuation, the system should mainly generate the even harmonics but the odd harmonics should not be suppressed. It follows from text above and from [5] as well that this can be achieved using the system with non-symmetrical signal limiting from Figure 7. A faster decrease in the amplitude of higher harmonics than with a classical limiter can according to (8) be achieved by increasing the slope of the transfer characteristics section, which performs the signal limiting. The generation of higher harmonics even for low amplitudes of the input signal can be achieved using a non-linear system with non-linearity around the operating point (see Figure 9).

Following equation describes the non-linear system designed for the distortion effects. The equation was designed according to characteristics of the broken-line approximation described above:

$$y[n]=\begin{cases} x[n](1-d_2)+x_T(d_1-d_2) & \text{for } x[n]\le -x_T \\ x[n](1-d_1) & \text{for } -x_T < x[n]\le 0 \\ x[n] & \text{for } 0 < x[n]\le x_T \\ x[n](1-d_2)+x_T d_2 & \text{for } x_T < x[n], \end{cases} \tag{17}$$

where $d_1, d_2 \in (0,1)$, x_T is the threshold level, d_1 is distortion ratio bellow this level and d_2 is the ratio above the threshold level.

In contrast to the common symmetrical limiters, the output signal of such system consists of the first and even harmonics only when amplitude of the harmonic input signal is bellow x_T. The distortion ratios above and bellow the threshold level can be adjusted almost independently. Higher harmonics with high dissonance ratio are attenuated when $d_1<0.1$.

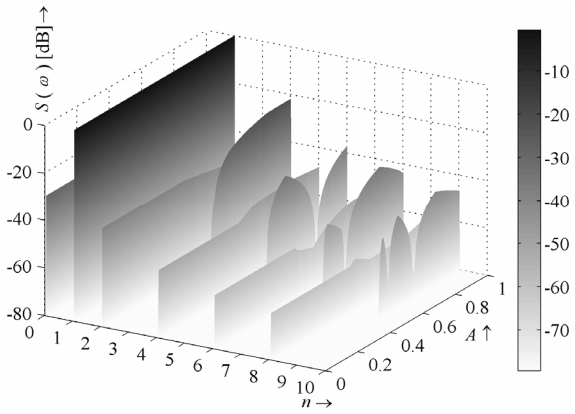


Figure 9: *Joint amplitude-frequency analysis of output signal of limiter with non-linearity around operating point.*

3. CONCLUSION

A direct realization of discrete non-linear systems with transfer characteristics approximated by broken-line function is controversial because of the aliasing distortion caused by the unlimited bandwidth of output signal of the system. However, we can easily modify the ratio of amplitudes of the output signal harmonics via simple changes of the approximation parameters while keeping the changes of the type of output signal spectrum under our control. The aliasing distortion can be suppressed using the input signal upsampling, whose ratio is determined using the psychoacoustical model, or we could find relations between the broken-line approximation parameters and the coefficients of its Taylor series. Future work will be focused on examining the output signal spectrum changes when transfer characteristic smoothing is used.

4. ACKNOWLEDGEMENTS

This paper is supported by project No. 102/06/1233 of the Grant Agency of the Czech Republic – Czech Science Foundation.

5. REFERENCES

[1] J. Schattschneider, U. Zolzer, "Discrete-time models for nonlinear audio systems". In *Proc. DAFX-99 Digital Audio Effects Workshop*, Trondheim, December 1999, pp. 45-48.

[2] J. Schimmel, "Audio Effect Synthesis Using Nonlinear Signal Processing", PhD. Thesis, BUT, 2006.

[3] E. O. Brigham, *The Fast Fourier Transform*. Prentice-Hall, Inc. 1984.

[4] F. Kouril, K. Vrba, Theory of Non-Linear and Parametric Circuits [in Czech]. SNTL Praha, 1981.

[5] U. Zölzer, *DAFX – Digital Audio Effects*, 1st ed. John Wiley & Sons, Ltd, 2002.

EFFECTIVE SINGING VOICE DETECTION IN POPULAR MUSIC USING ARMA FILTERING

Hanna Lukashevich, Matthias Gruhne and Christian Dittmar

Fraunhofer IDMT
Ilmenau, Germany
lkh@idmt.fraunhofer.de

ABSTRACT

Locating singing voice segments is essential for convenient indexing, browsing and retrieval large music archives and catalogues. Furthermore, it is beneficial for automatic music transcription and annotations. The approach described in this paper uses Mel-Frequency Cepstral Coefficients in conjunction with Gaussian Mixture Models for discriminating two classes of data (instrumental music and singing voice with music background). Due to imperfect classification behavior, the categorization without additional post-processing tends to alternate within a very short time span, whereas singing voice tends to be continuous for several frames. Thus, various tests have been performed to identify a suitable decision function and corresponding smoothing methods. Results are reported by comparing the performance of straightforward likelihood based classifications vs. postprocessing with an autoregressive moving average filtering method.

1. INTRODUCTION

The availability of digital music material to end users is continually increasing through new media and content distribution methods. As a result, there is a growing need to automatically categorize and annotate the large amount of data. This allows the user to locate music that fits his or her personal preferences. It's now common sense that semantically meaningful descriptions (e.g. genre, tempo and musical key) of audio content are a suitable means to achieve that goal.

Therefore, active research has been conducted in the field of Music Information Retrieval (MIR) during recent years. Discrimination between vocal and non-vocal parts of popular music has been identified as an importand base technology for further high-level analysis. This information can be used for example in artist identification [1] and singing language recognition [2]. It has furthermore much relevance in lyrics synchronization [3]. One of the early approaches of vocal/non-vocal detection in popular music has been derived from speech/music discrimination and introduced by Berenzweig and Ellis [4]. They performed experiments using several low-level descriptors and Hidden Markov Models (HMM) for discriminating between two classes of a previously annotated and trained database. The reported results vary between 55,2% and 81,2%, depending on the utilized features. Tzanetakis [5] performed experiments with different low-level features and a multitude of classifiers. The reported results range between 61% and 75%. Maddage et al. [6] introduced an approach for vocal/non-vocal detection without a previously trained classifier. They performed a Fourier transform on the subbands of the spectrum of the signal. Thereafter they decided if the signal is music or vocal based on simple thresholding. They reported an accuracy of 84%. Unfortunately all these approaches are not directly comparable, because all publications are based on a different test set, varying in musical content and size.

One of the base approaches that is relatively straightforward to implement uses Mel-Frequency Cepstral Coefficients (MFCCs) and a Gaussian Mixture Model (GMM) classifier. This technique has been used in artist detection, singing language detection and lyrics synchronization [1], [2], [3] and it exhibits performance comparable to more complex systems.

With the combination of MFCCs and GMMs one often encounters rapidly alternating output, that is semantically meaningless for the target application. Therefore, a smoothing function for decimation of outliers has been introduced in [7], where Tsai et al. accumulated the log likelihoods of single frames over a certain time span in order to achieve more reliable results. Thus, we decided to pursue this approach and concentrate on postprocessing of intermediate classification results. We identified that the instability in classifying depends on factors like model quality, generality of training data and complexity of test material. Since the influence of the above mentioned factors can only be reduced to a certain extent we investigated into finding a suitable smoothing algorithm. This paper introduces a novel method for deriving a bounded decision function and appropriate smoothing with an Autoregressive Moving Average (ARMA) filter [8].

The structure of the paper is organized as follows. The next two chapters describe feature vector extraction and GMMs. Section 4 presents our decision function, the subsequent ARMA filtering and additional smoothing. Thereafter the audio data set used in the evaluation is described. Section 6 depicts the details of the experiment and the corresponding results. Finally section 7 concludes this work and provides some perspectives for future directions.

2. FEATURE VECTOR EXTRACTION

From the multitude of features that have been suggested for MIR applications we have chosen to utilize MFCCs. MFCCs and derivates have found multiple successful applications in the field of speech recognition and speaker identification and has proved to be well-suited for MIR, for example in singer and artist identification [1], [2], [3]. The term cepstral originates from fundamental research of Bogert [9]. The main point is the implicit decomposition of a periodic signal into excitation and filter. The most straightforward way to compute MFCC is the summation of FFT bins weighted by the Mel-Filterbank passbands, taking the natural logarithm and subsequent discrete cosine transform.

The coefficients computed by that method can be thought of as

weighting factors for different periodic characteristics in the logarithmic distribution of energy in the Mel-bands. The very first coefficient equals the overall energy and should be omitted for classification purposes to be prone against different amplification factors. The succeeding coefficients represent a more detailed description of the energy distribution in Mel-bands. Therefore, the number of coefficients is limited to D in order to generalize the properties of the current audio frame whilst omitting subtile dynamic aspects. Furthermore, the implicit orthogonality of MFCCs simplifies the theoretical background of statistical modeling.

3. GAUSSIAN MIXTURE MODELS

Our main interest is targeted towards discrimination of the two classes: music and music plus singing voice (further denoted as MUS and VOX respectively). For each of the above mentioned classes one particular Gaussian mixture model represents the distribution of the raw data in D-dimensional feature space as linear combination of several D-dimensional Gaussian probability density functions (PDF). These two Gaussian mixture models are further denoted as MUS GMM and VOX GMM. The parameters of the component densities are estimated with the well-known expectation maximization (EM) algorithm [10] [11]. The linearly weighted combination of Gaussian basis functions is expected to generalize the collected features forming smooth approximations of their arbitrarily shaped PDFs. Equation 1 gives the definition of a GMM defined as a weighted sum of M component PDFs according to [12]

$$p(\mathbf{x}|\lambda) = \sum_{i=1}^{M} p_i g_i(\mathbf{x}) \tag{1}$$

where $g_i(\mathbf{x})$, $i = 1, ..., M$ represent the component PDFs, $\mathbf{x}$ is a D-dimensional observed feature vector and p_i the individual mixture weights or priors. Each component is defined as a D-variate Gaussian PDF

$$g(\mathbf{x}) = \frac{1}{(2\pi)^{D/2}|\mathbf{\Sigma}|^{1/2}}\, exp\left\{-\frac{1}{2}(\mathbf{x}-\mu)^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\mu)\right\} \tag{2}$$

with empirically estimated mean vector μ and covariance matrix $\mathbf{\Sigma}$. This way, a particular mixture PDF is completely parameterized by the tuple $\lambda_i = \{p_i, \mu_i, \mathbf{\Sigma}_i\}$. The training process is constituted by the maximum likelihood (ML) estimation of the model parameters that maximize the likelihood of the GMM given the training data consisting of feature vectors for one class. The ML optimization is actually carried out by the expectation maximization (EM) algorithm, iteratively refining the initial estimation of parameters [12].

The initial estimation of parameters is computed per model by choosing an appropriate M and partitioning the classes in feature space using k-means clustering. The preceding clustering step guarantees convergence to invariant ML estimates and is therefore favoured in contrast to random model initialisation.

4. DECISION FUNCTION

The usage of VOX GMM and MUS GMM allows us to calculate likelihoods of both models for every input frame of data. Let $L(\lambda_v|\mathbf{x})$ and $L(\mathbf{x}|\lambda_m)$ denote the likelihoods of feature vector $\mathbf{x}$, to belong to VOX and MUS classes respectively.

In previous works [7], a decision function was derived as a simple difference between log-likelihood values for VOX and MUS classes as given in equation (3).

$$f_1(\mathbf{x}) = log(L(\lambda_v|\mathbf{x})) - log(L(\lambda_m|\mathbf{x})) \tag{3}$$

If the value of the decision function is above the theoretical threshold of 0 then the corresponding frame is considered to belong to the VOX class while values below 0 indicate MUS class.

In this work we propose a novel approach for computing a decision function as given in equation (4).

$$f_2(\mathbf{x}) = \frac{L(\lambda_v|\mathbf{x})}{L(\lambda_v|\mathbf{x}) + L(\lambda_m|\mathbf{x})} - 0.5 \tag{4}$$

The theoretical threshold of the proposed decision function $f_2(\mathbf{x})$ is also arranged to 0. It should be noted that whithout further post-processing both decision functions essentially produce the same results, when it comes to a binary threshold based decision (i.e. indicate if $L(\lambda_v|\mathbf{x})$ is higher than $L(\lambda_m|\mathbf{x})$). Since both decision functions exhibit a very noisy slope, they are not directly suited for utilization in real-world applications. It is not beneficial to make a decision for audio excerpts that are too short to provide semantically meaningfull interpretations. Therefore, the decision functions need an additional smoothing and/or filtering.

Due to the complexity inherent to training two GMMs covering the entire body of real-world music, the absolute values of $L(\lambda_v|\mathbf{x})$ and $L(\lambda_m|\mathbf{x})$ tend to be relatively small. Moreover, absolute values of likelihoods for MUS and VOX parts even within a particular song may exhibit significant differences.

The statistical properties of the above mentioned decision functions have been examined in-depth in order to benefit from their peculiarities. We investigated the PDFs of the values returned by each of the desicion functions separately for MUS and VOX classes of input data. Since manually segmented songs from our audio data test set (see section 5) were available, we had the possibility to split the set of observed input feature vectors $\mathbf{x}$ in two subclasses: VOX and MUS frames contained in the song. For each of these subclasses the PDFs of the decision functions were estimated. Exemplary results for a representative song are shown in Figure 1. It can clearly be seen that although the experimental results for both decision functions proved the liability of the theoretical threshold, the PDFs do exhibit distinct properties. In the upper plot (results for $f_1(\mathbf{x})$), the overlapping region of the PDFs covers a large amount of observations. Thus, even small changes of the thresholding could have significant impact on the classification results. In contrast, the lower plot ($f_2(\mathbf{x})$) depicts overlapping in less critical regions. A well-established technique to improve correct classification rate is defining a so-called uncertain zone around the threshold. One can see that for the decision function $f_1(\mathbf{x})$ it will yield a high amount of uncertain frames. In addition, the borders of uncertain zone for $f_2(\mathbf{x})$ must be given in absolute values which tend to vary depending on the song.

Moreover, our experiments proved that the decision function $f_2(\mathbf{x})$ is the most suitable for filtering and smoothing. It is ranged between -0.5 and 0.5, and it is symmetrical around the threshold. As it will be shown below, $f_2(\mathbf{x})$ can be successfully filtered using ARMA filtering [8].

4.1. Autoregressive Moving Average Filtering

As singing voice generally tends to be continuous for multiple consecutive frames, we assumed that the instantaneous value of decision function of frame i is partly determined by k previous frames,

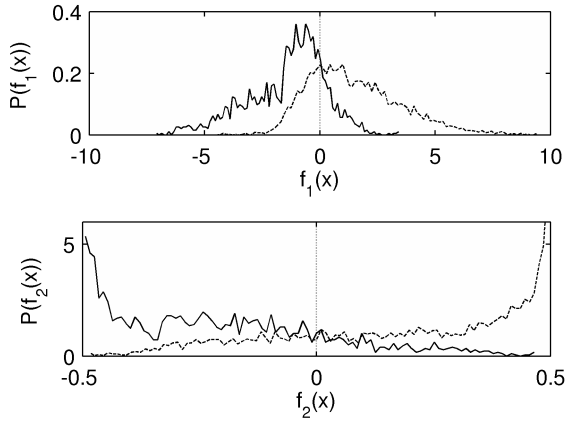


Figure 1: *Comparison of PDFs of the decision functions for a representative song. The solid line in both plots corresponds to MUS frames of the song, and the dashed line corresponds to VOX frames of the song. The upper plot shows the PDFs recieved for* $f_1(\mathbf{x})$ *and the lower plot shows the PDFs recieved for the decision function* $f_2(\mathbf{x})$.

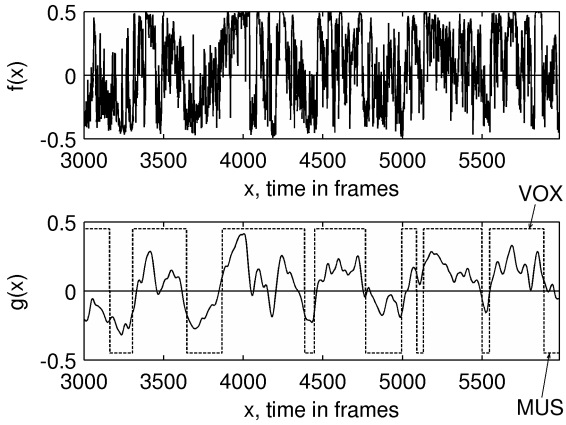


Figure 2: *The upper plot shows decision function* $f_2(\mathbf{x})$ *for an exemplary excerpt of a representative song. The lower plot depicts the decision function* $g(\mathbf{x})$ *after ARMA filtering and smoothing. For comparison, the dashed function represents the manual segmentation for this audio excerpt.*

i.e. it can be interpreted as autoregressive (AR) process. In addition, smoothing of the decision function for removing short term outliers can be efficiently performed by means of moving average (MA) processing. The combination of the above mentioned post-processing steps can be interpreted as an ARMA(*p*,*q*) process. This process can be approximated by a rational transfer function [13] given by the linear difference equation:

$$x_i = \sum_{l=1}^{q} b_l n_{i-l} - \sum_{k=0}^{p} a_k x_{i-k}. \tag{5}$$

The system transfer function $H(z)$ between the input (n_i) and the output (x_n) for the described ARMA process is the rational function $H(z) = B(z)/A(z)$, where $A(z)$ and $B(z)$ represent the z-transforms of the AR and MA branches respectively. We calculate the coefficients b_l and a_k of the ARMA filter via Prony's method [13], [14]. Prony's method is an algorithm for finding an IIR filter with a prescribed time domain impulse response. This filter can recover the coefficients b_l and a_k exactly if the data sequence is truly an ARMA process of the correct order. The order of the ARMA filter was determined experimentally. The best results were received for $p = q = 10$. An examplary result of ARMA filtering applied to the decision function $f_2(\mathbf{x})$ is shown in the lower plot of Figure 2. In that plot, additional smoothing via convolution with a Hamming window was applied.

5. AUDIO DATA TEST SET

To assess the performance of the proposed method, we had to define a proper evaluation test bed. Due to the fact that there exists no well established database for that particular task, we decided to set up a proprietary test set by ourselves. Our test database consists of 84 PCM WAV-files. All files are downsampled to 16 bit, 22050 kHz, mono. The database contains 10 singers: 5 male and 5 female (see Table 1). The songs of every singer were randomly separated into training set (38 songs, 3-5 songs for every singer) and test set (46 songs, 4-5 songs for every singer). Every record in the database was manually labeled with regard to instrumental and vocal parts using the open source tool Wavesurfer. The total duration of the training set is 5815.47 sec, which equals more than 1.5 hours of music. The total duration of the test set is about 3000 sec, or 50 minutes, whereas only 1 minute excerpts of every song were considered (from 20 sec to 80 sec).

Male Singers	Female Singers
Brian Adams	Barbara Streisand
Eros Ramazotti	Anna Netrebko
Frank Sinatra	Nelly Furtado
Ozzi Osbourne	Anne Clark
Sting	LeAnn Rimes

Table 1: Singers in the Database

6. EVALUATION AND RESULTS

At the stage of feature extraction we used $D = 13$ Mel-frequency coefficients computed with 30 ms framesize and 10 ms hopsize. As our system is considered to constitute a front-end for further singer identification and lyrics alignment, we focussed on minimizing the error in identification of MUS frames, thus errors for VOX frames were considered to be less critical. For that reason, the number of mixtures for MUS GMM was set to 20, and the number of mixtures for VOX GMM was 13. These optimum parameters had been identified experimentally, the search was performed in an interval from 4 to 52 mixtures per model. The covariance matrices Σ were assumed as diagonal, considering the fact that they describe uncorrelated MFCCs.

The criterion F used to describe the classification rate has been defined as the harmonic mean (8) of V (6) and M (7).

$$V = \frac{number\ of\ voice\ frames\ detected\ correctly}{total\ number\ of\ voice\ frames} \tag{6}$$

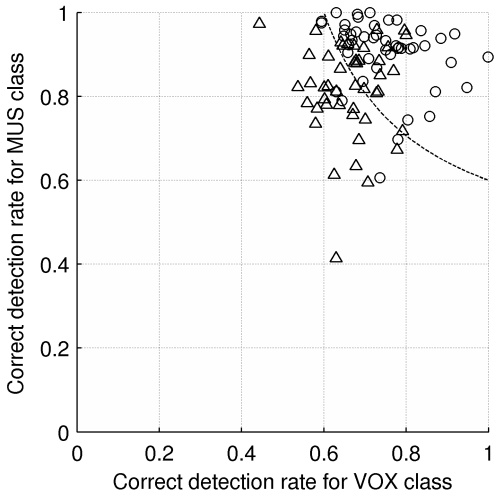


Figure 3: *Correct detection rates for all 46 songs of the test set. Dotted line corresponds to $F = 0.75$. Triangles represent baseline classification results without post-processing. Circles depict classification results achieved with the proposed method.*

$$M = \frac{number\ of\ music\ frames\ detected\ correctly}{total\ number\ of\ music\ frames} \quad (7)$$

$$F = \frac{2VM}{V+M} \quad (8)$$

Figure 3 shows the achieved results for each of the 46 songs of the test set with and without post-processing. Application of the proposed approach resulted in an average increase of the F-score from 72,7% to 81,3%. With our approach, the average result for MUS class is 90,5% while the average result for VOX class is 75,0%. Classification performance can be observed to increase significantly for the VOX class. This is due to fact that the VOX GMM contains less mixtures than its counterpart. So the possibility of spurious thresholding becomes higher in the raw unsmoothed detection function. As we mentioned before, the mistakes in the MUS class are considered more critical and therefore the outcomes correspond to our target. Besides relatively high correct detection rate, the usage of the suggested approach allows to retrieve semantically meaningful consecutive song segments of MUS and VOX. These can be effectively used for further applications e.g. lyrics alignment.

7. CONCLUSION

This paper described our approach towards automatic detection of singing parts in popular music. We used the well established methods of combining MFCCs and GMMs as a front-end. We showed that comparably straightforward methods of post-processing produce significant increase in classification results. Moreover, the application of the proposed decision function in conjunction with subsequent ARMA filtering explicitly enhances the perceptual quality of the achievable song segmentation. The properties of the described decision function can presumably be exploited in systems using further audio features and additional classification techniques such as HMMs, Support Vector Machines or Neural Networks. The information that can be derived from statistical analysis of the decision function allows for additional refinement stage based on heuristics. In addition, the filtered and smoothed decision function carries valuable information that can be interpreted in a semantically meaningful manner. For instance, its local minima indicate borders of phrases apparent while singing. These peculiarities will be studied more in-depth in future work.

8. REFERENCES

[1] A. Berenzweig et al., "Using voice segments to improve artist classifiaction of music," in *Proc. AES 22 International Conference on Virtual, Synthetic and Entertainment Audio, Espoo, Finland*, June 2002.

[2] W. H. Tsai and H.-M. Wang, "Towards automatic identification of singing language in popular music recordings," in *Proc. of the 5th International Conference on Music Information Retrieval (ISMIR)*, 2004.

[3] S. G. Kai Chen, "Popular song and lyrics synchronisation and ist application to music," in *Proc. of the Thirteenth Annual Conference on Multimedia Computing and Networking (MMCN)*, 2006.

[4] A. Berenzweig and D. Ellis, "Locating singing voice segments within music signals," in *Proc. of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, October 2001.

[5] G. Tzanetakis, "Song-specific bootstrapping of singing voice structure," in *Proc. of the IEEE International Conference on Multimedia & Expo (ICME)*. 2004, pp. 2027–2030, IEEE.

[6] N. Ch. Maddage, K. Wan., Ch. Xu, and Y. Wang, "Singing voice detection using twice-iterated composite fourier transform," in *Proc. of the IEEE International Conference on Multimedia & Expo (ICME)*. 2004, pp. 1347–1350, IEEE.

[7] W. H. Tsai and H.-M. Wang, "Automatic singer recognition of popular music recordings via estimation and modeling of solo vocal signals," *IEEE Transactions on Audio, Speech & Language Processing*, vol. 14, no. 1, pp. 330–341, 2006.

[8] S. L. Marple, Jr., *Digital spectral analysis with applications*, Prentice Hall, Englewood Cliffs, 1987.

[9] B. P. Bogert, M. J. R. Healy, and J. W. Tukey, "The frequency analysis of time series for echoes: cepstrum, pseudo-autocovariance, cross-cepstrum, and saphe cracking," in *Proc. of the Symposium on Time Series Analysis, Ed.: M. Rosenblatt, John Wiley, New York*, 1963, pp. 209–243.

[10] T. K. Moon, "The expectation-maximization algorithm," *IEEE Signal Processing Magazine*, vol. 13, no. 6, pp. 47–60, November 1996.

[11] N. M. Dempster, A. P. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. R. Statist. Soc. B*, vol. 39, pp. 185–197, 1977.

[12] D. A. Reynolds and R. C. Rose, "Robust text-independent speaker identification using gaussian mixture speaker models," *IEEE Trans. on Speech and Audio Processing*, vol. 3, no. 1, pp. 72–83, 1995.

[13] S. L. Marple, "A tutorial overview of modern spectral estimation," in *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1989, pp. 2152–2157.

[14] T. W. Parks and C. S. Burrus, Eds., *Digital Filter Design*, J. Wiley & Sons. Inc., 1987.

NON-LINEAR DIGITAL IMPLEMENTATION OF A PARAMETRIC ANALOG TUBE GROUND CATHODE AMPLIFIER

Francesco Santagata, Augusto Sarti, Stefano Tubaro

Image and Sound Processing Group
Dipartimento di Elettronica e Informazione
Politecnico di Milano, Piazza L. Da Vinci 32 - 20133 Milano - Italy
santagata/sarti/tubaro@elet.polimi.it

ABSTRACT

In this paper we propose a digital simulation of an analog amplifier circuit based on a grounded-cathode amplifier with parametric tube model. The time-domain solution enables the online valve model substitution and zero-latency changes in polarization parameters. The implementation also allows the user to match various types of tube processing features.

1. INTRODUCTION

Valves are today mainly limited to musical analog processors such as stomp boxes, equalizers, dynamic processors, power amplifiers, etc. As a matter of fact they are physically used in commercial devices because their performance and sound are generally quite difficult to match with digital processing systems [1]. Although in recent years digital processors have gained more and more respect in this field, musicians are still reluctant to give up the "warmth" and the "added dirt" that make the tube sound so characteristic. Our DSP solution for digital digital tube simulation in ground cathode configuration is based on Koren's phenomenological tube model [2], which has the ability to match different harmonic distributions and dynamic behavior. This solution was chosen over other triode tube models for its flexibility and its intuitive parametrization.

The common cathode circuit was analyzed and split between polarization circuit and small signal circuit even if the solution is calculated on the large signal. In fact, the tube has a "built-in feedback", as a large signal on the grid affects the gain of the circuit: the tube polarization that sets the desired voltage gain is, in fact, affected by the same amplified input signal that is present on the plate. More recently a real-time wave digital solution was presented [3], which discusses a different time-domain technique to solve the same problem.

In this paper we discuss the reference analog circuit, the polarization and the small signal solution, and how they are combined with the nonlinear resistance. After examining the dynamic properties of the stage, we finally present the adopted solution.

2. GROUNDED CATHODE TUBE AMPLIFIER

The circuit consists of three voltage generators: V_1 is the DC power supply, V_2 is the AC or DC heater supply used for warming the tube up, V_{in} is the input signal. The 12AX7 tube polarization is set by three resistances: R_1 is the anode resistor, R_2 is the cathode resistor, R_3 is the load resistor. V_{in} is directly connected to the grid. While increasing the grid voltage, the current in the tube that flows from the anode (plate) to the cathode increases, which means that the voltage on R_1 increases as well. As V_1 is fixed, the voltage on R_3 decreases, and so does the corresponding current (which is small compared to R_3 polarization current). The situation when we decrease the grid voltage is completely dual, except for the fact that the relationship between grid voltage and plate current is non linear.

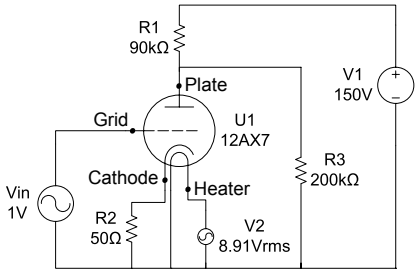


Figure 1: *Simplified grounded cathode triode amplifier schematic.*

3. PHENOMENOLOGICAL VALVE MODEL

According to [4], there are three aspects to consider in modeling and simulating the nonlinear behavior of valves:

- the stage is expected to work at audio frequency, therefore secondary effects such as Miller capacity are less than relevant;
- the heater does not need to be included in the model as it is independent from the grid, plate and cathode signals;
- the model is expected to be as general as possible, so that different tube models can be accommodated within its parameter space.

The tube input (grid) will be here considered as an ideal voltage reader (infinite impedance port), which means that the polarization can be passed to the model as a parameter. Any voltage offsets, due to the fact that a small current is flowing into the grid (which has a finite impedance) will be compensated by another parameter. It is thus quite reasonable to model only the nonlinear current generator in the output section. This allows us to focus our attention on the dynamic plate-cathode resistor. The phenomenological model turns out to be particularly well-suited for describing the real nonlinear function of the triode and for matching different types of tube models.

3.1. Koren's Triode Model

Koren [2] proposes a phenomenological model that is very close to the expected triode behavior on a wide range of plate currents and voltage values. This model is based on the triode 3/2 power law and it consists in the two following equations:

$$E_1 = \frac{V_{pk}}{K_p} \cdot \log\left(1 + \exp\left(K_p \cdot (\frac{1}{\mu} + \frac{V_{gk} + V_{ct}}{\sqrt{K_{vb} + V_{pk}^2}})\right)\right), \quad (1)$$

$$I_p = \frac{E_1^{Ex}}{K_{g1}} \cdot \big(1 + sgn(E_1)\big), \quad (2)$$

where V_{pk} is the plate cathode voltage, V_{gk} is the grid cathode voltage and I_p is the plate current. An extensive description of the model and of its parameters μ, K_{g1}, Ex, K_p, V_{ct}, K_{vb} can be found in [5] and [6]. V_{ct} is the contact potential between grid and cathode, which can be seen as an offset on the grid voltage:

$$V'_{gk} = V_{ct} + V_{gk}, \quad (3)$$

Eq. (3) enables a more accurate matching of the plate I-V characteristic, with the result of improving the accuracy of the parametrization. Ex and K_{g1} can be optimized to obtain a good matching with the experimental data for low grid voltage. K_p models the behavior for large negative grid values while K_{vb} is related to the location of the "knee" of the plate curve.

TUBE	μ	K_{g1}	K_p	K_{vb}	V_{ct}	Ex
12AX7	100.8	1890	828	72	0.612	1.4979
ECC88	32.92	155.625	225	4492	0.248	1.204
300B	3.99	2715	51	3.9375	2.16	1.526

Table 1: *Parameters of different tubes.*

4. CIRCUIT CONSIDERATIONS AND SOLUTION

The system is modeled in a rather traditional fashion: the circuit is split into a linear polarization circuit and a small-signal nonlinear circuit.

4.1. Polarization Circuit

The plate current I_p is computed with a fixed E_p (linearization around the selected working point). There are four resistors: the anode resistor R_a, the cathode resistor R_k, the load resistor R_l, and the static plate-cathode estimated resistance. R_a, R_k, V_{aa} and E_p are user-controlled parameters. R_l exists if:

$$V_{aa} < E_p + I_p \cdot (R_k + R_a). \quad (4)$$

R_l and R can be computed as in eqs. (5) and (6):

$$R = \frac{E_p}{I_p}, \quad (5)$$

$$R_l = \frac{E_p + I_p \cdot R_k}{\frac{V_{aa} - E_p + I_p \cdot R_k}{R_a} - I_p}, \quad (6)$$

which preserves the validity of eqs. (1), (2). This circuit is rather flexible, as it computes the adapted load on a desired polarization and decouples input from output. The input port is an ideal voltage reader, with a bias that sets the desired working point of the triode. Other currents are computed with current partitioning resistor-sets.

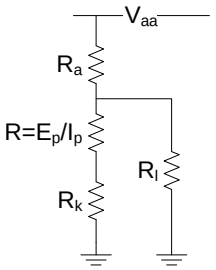


Figure 2: *Polarization circuit schematic.*

4.2. Small-Signal Circuit

The small-signal circuit in Fig. 3 is composed of three static resistors, a non linear resistor, and an ideal nonlinear voltage-controlled current generator. R_a, R_k and R_l are the same as those of the polarization circuit, while i_p and r must be considered in a different way. Notice that Koren's model is a large signal model therefore every current computation is based on the sum of small signal and a polarization voltage.

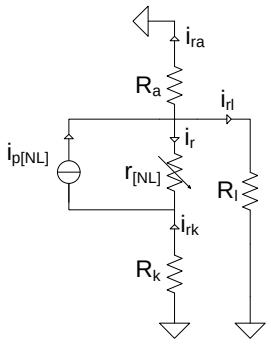


Figure 3: *Small-signal circuit schematic. Orientation of currents for a negative input, $v_g < 0$*

The current generator injects a negative current into the circuit when a positive voltage v_g is applied to the grid. The tube gain is defined by the model parameters but also by the voltage between anode and cathode U_p and by the large-signal voltage that drives the grid U_g. The current is split in the node on the right of the generator by a current divider: i_r controlled by i_p is the small-signal current in the dynamic resistor between anode and cathode. As a consequence we have

$$i_r = i_r\big(i_p, r(i_p)\big), \quad (7)$$

$$v_p = i_r \cdot r = i_r\big(i_p, r(i_p)\big) \cdot r(i_p), \quad (8)$$

where v_p is the small-signal contribution to the plate polarization. The current divider equations allow us to compute all voltages in the circuit.

4.3. Non Linear Dynamic Resistor

On the basis of experimental data taken from a 12AX7 tube datasheet we found that with a plate cathode voltage of $250V$ and plate current of $1.2mA$ the plate resistance is close to $210k\Omega$: this value is

quite different from the $62.5k\Omega$ that appears on the datasheet.

$$r = a \cdot exp(b \cdot J_p) + c \cdot exp(d \cdot J_p) \tag{9}$$

Using a suitable model and curve fitting it is possible to compute the dynamic resistance as a function of the plate's large signal J_p.

coefficient	estimated value	95% confidence interval
a	$3.268 \cdot 10^5$	$(3.164 \cdot 10^5, 3.373 \cdot 10^5)$
b	-5238	$(-5347, -5129)$
c	$9.174 \cdot 10^4$	$(9.078 \cdot 10^5, 9.27 \cdot 10^5)$
d	-315.3	$(-322.7, -307.9)$

Table 2: *Values that model the nonlinear current variable plate resistance of a 12AX7 tube with* $250V$ *on the plate.*

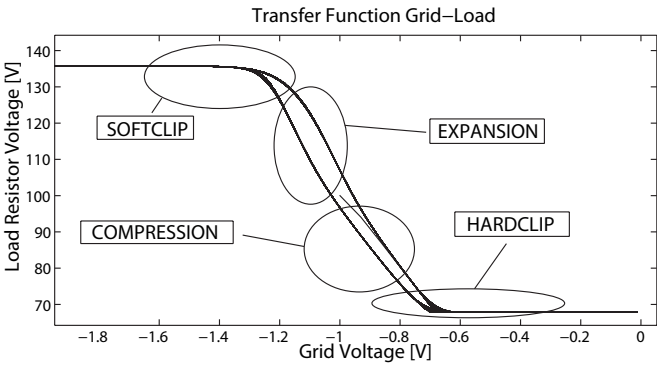


Figure 4: *Cyclic behavior of transfer function's working conditions, caused by a 2V sine wave input*

4.4. Large-Signal Operation

Large signals are computed with the sign convention that current is positive flowing top to bottom in Fig. 3. Eqs. (1) and (2) can be used for computing J_p as a function of U_g and U_p.

$$U_p = E_p + v_p, \tag{10}$$

With a fixed grid voltage large signal value U_g we have

$$U_p = U_p(i_p, r) = U_p\Big(i_p(U_p), r\big(i_p(U_p)\big)\Big). \tag{11}$$

Eq. (9) will use J_p as variable which is the maximum current which flows in the tube. With reference to eqs. (1) and (2) it is possible to notice a certain asymmetry with respect to the working point.

$$i_p\big(U_p, v_g\big) > \big|i_p\big(U_p, v_g'\big)\big| \qquad \forall v_g > 0, \forall v_g' < 0, \tag{12}$$

i_r is computed by a current divider between i_p and r to simulate an approximation of the nonlinear behavior of r in parallel configuration with a voltage-controlled current generator. We found that a small-signal voltage v_p between plate and cathode was large enough to affect E_p, which plays a significant role in the built-in tube compression.

4.5. Working Conditions

One of the aspects that make the valve behavior musically interesting is their "gentle" dynamic compression. The input signal is transferred to the output with a modified amplitude ratio: small voltages are more amplified than big values and, taking this to the extreme, signal may be affected first by soft clipping and then by hard clipping when the load reaches the voltage imposed by the power supply. Unlike what expected from a "smooth" sounding device, even triode amplifiers are capable of hard clippings [7].

With the current signs of Fig. 3 a negative input enables a larger current flow than a zero input because of the contribution of $i_r \cdot r$ to the plate voltage, which increases the tube gain. The input signal turns out to be more expanded until the magnitude of the negative input becomes large enough. On the other hand reducing the input too much blocks the flow of currents. Soft clipping depends on the shapes of the tube transfer function, as negative inputs are subject to gain expansion and then gradually to heavier compression.

As far as compression is concerned, it is useful to think of currents in Fig. 3 with inverted signs, which allows us to deal with positive signal with respect to grid polarization. Voltage contribution $i_r \cdot r$ is negative and is lowering the plate cathode voltage giving rise to gain reduction. If the input value gets too high (when i_{rk} gets equal to the polarization current flowing in R_k), the system starts saturating until the large-signal voltage on R_a goes to zero and the system begins hard clipping the output to a voltage value that is fixed by the power supply.

5. NUMERICAL SOLUTION

The algorithm consists basically of two parts: polarization and small-signal output computation. First the following polarization variables are set: E_g, E_p, R_a, R_k and V_{aa}. On the basis of these parameters plate current $I_p\big(E_g, E_p\big)$, static resistance R and dynamic resistance for null input $r_0(I_p)$ are computed. Then the load resistor $R_l\big(E_p, I_p, R_a, R_k\big)$ (if it exists with the desired values) is computed with an imposed plate cathode voltage. In the small-signal solution the voltage grid input is updated:

$$U_g = E_g + \frac{v_g}{C}, \tag{13}$$

where C is a scaling factor. The plate polarization E_p is updated with the small-signal voltage contribution of the previous sample: this will be clearer with eq. (17). Large signal $J_p\big(U_g, U_p\big)$ is computed by using the plate polarization state of the previous sample.

$$U_p(n) = E_p + v_p(n-1), \tag{14}$$

This voltage influences the gain of the stage, therefore the dynamic resistor for that plate current is computed. The saturation takes place when the small signal i_p cancels the large signal on R_k and then on R_a. In both cases the growth of $|i_p|$ is stopped and a zero voltage ends up being forced on both resistors: these conditions depend on polarization currents. If i_p turns out to be saturated, the large signal J_{pS} and the dynamic resistor $r_{1S}(J_{pS})$ are recomputed. A good estimation of plate current and dynamic resistance based on that polarization are determined. A new estimated current $i_r(i_p)$ can be obtained from the value of J_p computed in the previous step. In fact, i_r multiplied by r_1 or r_{1S} returns a good

estimation of the plate polarization.

$$U_{p_1}(n) = E_p + i_r(n) \cdot r_1(n) \tag{15}$$
$$= E_p + i_r\Big(i_p\Big(v_g(n), v_p(n-1)\Big), r_1(n)\Big) \cdot r_1\Big(J_p(n)\Big).$$

The value of U_{p_1} is kept as a safe estimate of the tube's gain while a new "final" estimation of both plate current $J_{p_X}(U_g, U_{p_1})$ and dynamic resistance $r_2(J_{p_X})$ are computed. Through a new evaluation of the working conditions we can find whether the tube is saturating and fix the current at $i_{p_X S}$ and the dynamic resistor at $r_{2S}(i_{p_X S})$. With the second estimation of the plate current it is possible to calculate the output voltage:

$$v_{out} = V_{aa} - U_{ra} - I_{rl} \cdot R_l. \tag{16}$$

As a final step, the plate polarization based on J_{p_X} is sent to the next sample

$$v_{pF}(n) = v_p(n+1) = r_2(i_{p_X}) \cdot i_r(i_{p_X}, r_2). \tag{17}$$

The plate's small signal contribution is recursively used for the computation of the next sample.

6. RESULTS

Building a digital model of an analog circuits enables to have different dynamic and frequency response even from the same tube, or better, from the same nonlinearity. The filter is working at $192kHz$ to avoid aliasing problems.

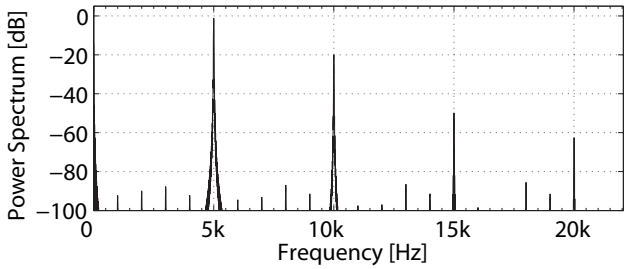


Figure 5: *12AX7 soft driven with 5kHz sine input*

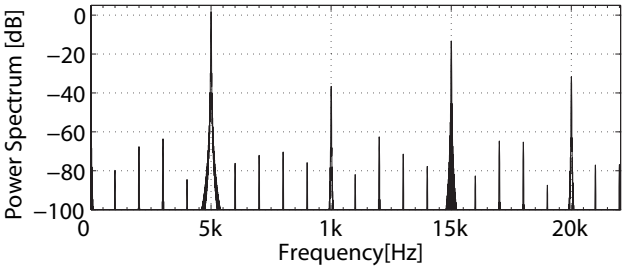


Figure 6: *12AX7 hard driven with 5kHz sine input*

In Fig. 5 input gain and R_a are smaller than in Figure 6 but the tube is the same. The aliased components are slightly greater than in [3] but this stage implements hardclipping. Different effects can be produced by reducing the power supply voltage as output

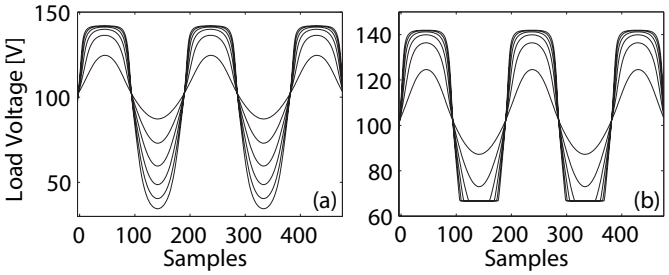


Figure 7: *12AX7 output for 1kHz sinewave input: voltage ranges from 0.25V to 1.5V in 0.25V steps. (a) hard clipping routine disabled, (b) hard clipping routine enabled*

waveform is pushed down in the negative half because amplifier dynamic is reduced. By changing the working point, E_g and E_p, the model produces different distributions of harmonics.

Fig. 7 shows a comparison with the stage distortion in [3]. The aliasing is still acceptable even when causing the stage to go into hard clipping. Reducing the sampling rate at $44.1kHz$ allows only soft clip operation with an acceptable $15dB$ signal degradation: hardclipping involves harsh sounding alias. The adopted model does not account for capacitive behavior and grid input model.

A correct parametrization for large positive grid voltage is yet to be found. Real-time parameter adjustments would improve the performance and the ease of use.

7. CONCLUSIONS

We proposed a versatile nonlinear processing stage for tube simulation that allows the user to account for a variety of distortions, from mild even harmonics to heavier odd harmonics. A specific class of tube models was tested and proved to provide a dynamic spectral response.

8. REFERENCES

[1] E. Barbour, "The cool sound of tubes," *IEEE Spectrum*, pp. 24–32, August 1998.

[2] N. Koren, "Improved vacuum-tube models for spice simulations," *Glass Audio*, vol. 8, no. 5, pp. 18–27, 1996.

[3] J. Pakarinen M. Karjalainen, "Wave digital simulation of a vacuum-tube amplifier," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'06)*, May 2006.

[4] T. Serafini, "A complete model of a tube amplifier stage," `http://www.simulanalog.org/`, 2004.

[5] D. Nizhegorodov, "Model paint tools: Trace tube parameters over plate curves, interactively," `http://www.geocities.com/dmitrynizh/tubeparams_image.htm`.

[6] S. Perugini, "Vacuum diode models and pspice simulations," *Glass Audio*, vol. 10, 1998.

[7] R. O. Hamm, "Tubes vs transistors: Is there an audible difference?," *Journal of the Audio Engineering Society*, May 1973.

A SIMILARITY MEASURE FOR AUDIO QUERY BY EXAMPLE BASED ON PERCEPTUAL CODING AND COMPRESSION

Marko Helén and Tuomas Virtanen

Tampere University of Technology
Institute of Signal Processing
Korkeakoulunkatu 1, FIN-33720 Tampere, Finland
marko.helen@tut.fi

ABSTRACT

Query by example for multimedia signals aims at automatic retrieval of samples from the media database similar to a user-provided example. This paper proposes a similarity measure for query by example of audio signals. The method first represents audio signals using perceptual audio coding and second estimates the similarity of two signals from the advantage gained by compressing the files together in comparison to compressing them individually. Signals which benefit most from compressing together are considered similar. The low bit rate perceptual audio coding preprocessing effectively retains perceptually important features while quantizing the signals so that identical codewords appear, allowing further inter-signal compression. The advantage of the proposed similarity measure is that it is parameter-free, thus it is easy to apply in wide range of tasks. Furthermore, users' expectations do not affect the results like they do in parameter-laden algorithms. A comparison was made against the other query by example methods and simulation results reveal that the proposed method gives competitive results against the other methods.

1. INTRODUCTION

The management of ever growing multimedia databases is very time consuming when done completely manually. This is why automatic systems are required to lighten the job. Query by example aims at automatical retrieval of samples from a database, which are similar to a user-provided example. For example, a user gives an example of a dog barking and the system returns all the samples from the database which contain dog barking.

The concept of similarity itself is very problematic. Measuring similarity of audio samples without annotations is very difficult comparing to a text-based search, since the similarity in signal level does not correlate to human's impression of similarity. For example in the situation when there is an example of male speech, it is impossible to know whether the user wants samples from the same speaker, or about the same topic.

Most of the existing audio query by example systems approach the problem as follows. first, features from the example signal and from the database signals are extracted. Second, the distance between the example signal and each database signal is estimated. Finally, the samples which have the shortest distance to the example are retrieved.

Pampalk estimated a Gaussian mixture model (GMM) for the example and estimated the similarity by the likelihood that the database sample was generated by this model [1]. Mandel and Ellis [2] calculated the mean of each feature over the whole sample and used the Mahalanobis distance between the samples as a similarity measure. They also used the Kullback-Leibler divergence between two GMMs to estimate the similarity.

Helén and Lahti [3] used a histogram based method, which generated feature histograms for each signal, and calculated the distances between these histograms. They also used a method, which generates hidden Markov model (HMM) for each sample and also a universal background model using the whole database. Then they estimate whether it is more likely for the database signal to be generated by the example HMM or the background model. Helén and Virtanen proposed a method for estimating similarity by calculating the Euclidean distance between two GMMs of the features [4].

When measuring the similarity between two samples, parameters like the feature set have to be decided a priori. The choice of these parameters is crucial for the results and choosing the right parameters requires a lot of knowledge about the specific task. As a consequence, algorithm developer's expectations and presumptions have an effect on the results. It would be profitable to have a similarity metric that is not dependent on the user.

The proposed method utilizes low bit rate audio coding, which retains the perceptually most relevant information of the signal. The similarity of two samples is estimated using compression based similarity measure. The proposed method does not require setting of any parameters and it is especially practical in applications where there is very little knowledge about the contents of the database beforehand.

The paper is organized as follows. Section 2 describes the overview of the system, Section 3 describes the signal representation, Section 4 presents the compression based similarity metric. Section 5 gives experimental results and comparisons to the other methods and finally Section 6 is for conclusions.

2. SYSTEM OVERVIEW

The overview of the system is illustrated in Fig. 1. First, perceptual audio coding (MP3, AAC etc) is applied to the original audio files. Second, the coded signals are compressed alone using some lossless compression method (gzip,bzip etc.). Third, the files are concatenated into a single file and compressed together using the same compression method. Finally, similarity is calculated by estimating the benefit achieved by compressing the files together.

When the similarity estimates are received, there are two application-dependent main possibilities how to return the results to the user. The first, referred as k-nearest neighbor query (k-NN) [5], is to sort the signals in order of the similarity and retrieve a fixed number of most similar samples to the user. A drawback is

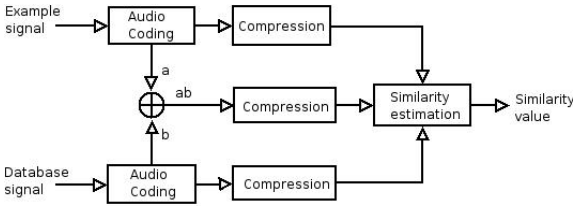


Figure 1: *Overview of the similarity estimation.*

that there is a possibility that some of the received samples are very different from the example, since a fixed number of samples is retrieved. Furthermore, the whole database have to be queried before the results can be presented.

The other possibility is to set a threshold, and retrieve all the samples that are closer than the threshold. This method is referred as ϵ-range query [5]. All samples inside the ϵ neighborhood of the query sample are retrieved. This way all the retrieved samples should be relatively similar to the example and similar samples may already be returned to the user during the query. The disadvantage of this method is that adjusting the threshold may not be straightforward and it might require user feedback or going through the whole database. In this study both methods are considered.

3. SIGNAL REPRESENTATION

The compression-based similarity measure requires a representation, where similar signals contain identical parts. A digital PCM signals are therefore too precise for this purpose. Perceptual audio coding provides a representation, where perceptually most important characteristics of a signal are retained and the signal is quantized so that identical codewords are present.

3.1. Perceptual audio codecs

Perceptual audio coding aims at representing an audio signal with a small amount of data while retaining the perceptual quality as close to the original as possible. Contrary to source coding, generic audio codecs remove the data which is perceptually irrelevant [6, pp. 41-42], thus they are lossy. They achieve compression by utilizing the properties of the human auditory system, especially the masking phenomenom. It refers to a situation where a separately audible sound becomes inaudible in the presence of a louder sound. The phenomenom is strong when the sounds occur simultaneously and are closely spaced in frequency.

General-purpose perceptual audio codecs are currently widely used in consumer electronics, for example in digital television, internet audio, and portable audio devices. The most commonly used codecs are developed in the standardization framework of Moving Picture Experts Group (MPEG). They include MPEG-1 Layer 3 (commonly known as MP3) and its successor Advanced Audio Coding (AAC). The perceptual codecs tested in this system include MP3 encoder LAME[1] and AAC encoder FAAC[2].

The basic idea of perceptual audio codecs is to quantize the input signal so that the quantization noise is inaudible. Since the masking phenomenon can be more easily modeled in time-frequency domain, codecs calculate a time-frequency representation using a filter bank or short-time frequency transforms. An auditory model approximates the masking effect, measures the audibility of the quantization noise, and controls the amount of bits required to represent the signal. The redundancy of the quantized codewords can be reduced by entropy coding.

[1] `http://lame.sourceforge.net/`

[2] `http://www.audiocoding.com/`

4. SIMILARITY MEASURE

To measure the similarity, we apply a measure developed by Bennett et al., which approximates the information distance between two sequences by compression [7]. The similarity measure has been previously used to a wide range of tasks: fetal heart rate tracings [8], classification of books by the author, optical character recognition, and building an evolutionary tree from mitochondrial genomes [9]. These studies show that the measure can be used in a wide range of application areas, and it does not need any specific knowledge about the task. Accuracy of such parameter-free algorithm is shown to be superior compared to traditional methods [10]. The distance used is referred as normalized compression distance, which is an estimate of normalized information distance.

4.1. Normalized compression distance

The minimum amount of information required to represent given string x is refered as Kolmogorov complexity. $K(x|y)$ is the conditional Kolmogorov complexity of string x relative to string y defined as the length of the shortest binary program to compute x if y is given as an auxiliary input. The minimum amount of information required to generate string x from string y and vice versa is refered as information distance (ID) [7]:

$$ID(x,y) = max\{K(x|y), K(y|x)\}. \tag{1}$$

This distance metric has two major drawbacks. First, it measures absolute distances meaning that two short random samples would have the same distance as two, almost similar, long samples. In order to have relative distance metric, the normalized information distance (NID) was proposed in [11]:

$$\mathrm{NID}(x,y) = \frac{max\{K(x|y), K(y|x)\}}{max\{K(x), K(y)\}}, \tag{2}$$

The other drawback is that this distance metric is based on the notions of Kolmogorov complexities, which are noncomputable. As a consequence, the approximation of the metric has to be used. The $K(x)$ and $K(y)$ are approximated here using $C(x)$ and $C(y)$, which are the sizes of compressed x and y respectively. The similarity between two signals is therefore approximated using a normalized compression distance (NCD) [9]:

$$\mathrm{NCD}(x,y) = \frac{C(xy) - min\{C(x), C(y)\}}{max\{C(x), C(y)\}}, \tag{3}$$

where $C(xy)$ is the compressed size of concatenated x and y. NCD is the measure of difference, thus larger values stand for more different signals. The value of NCD is between 0 and $1 + \epsilon$, since the compression techniques are not ideal.

This method can also be seen as parametric method, because the compression algorithm has to be chosen. However, the objective is to get the best approximation of the Kolmogorov complexity, therefore the algorithm that provides the best compression ratio should be chosen.

5. SIMULATION EXPERIMENTS

The performance of the proposed system was tested against other query by example methods. The methods were the Euclidean distance between the GMM densities [4], likelihood of GMMs [1], feature histogram based method [3], KL divergence of one-component GMMs [2], and Mahalanobis distance of feature means [2].

All these methods use the following preprocessing: first, signals are divided into 46 ms frames and second, several features are extracted from the frames. The feature set used here is the same as in [3] and [4]: Mel-frequency cepstral coefficients (three first coefficients), spectral spread, spectral flux, harmonic ratio, maximum autocorrelation lag, crest factor, noise likeness, crest factor, total energy, and variance of instantanous power. Before the processing, each feature is normalized to have zero mean and unity variance over the whole database.

Tested audio coding methods were MP3, AAC, and adaptive multi-rate (AMR). Bitrates between 8-64 kbits/s were tested and the best ones were chosen to be presented here. In AAC we used a version which does not apply frame wise Huffman coding to the signal, because this gave slightly better results than the original one. The method was also tested directly to wave files without any perceptual audio codec. Different lossless compression algorithms were also tested but the results were almost the same for all of them, the gzip is used in the simulations.

Simulations were carried out using an audio database which contains 1332 samples with 16 kHz sampling rate. The signals were manually annotated into 4 main classes and 17 sub classes. The classes and the number of samples in each class are listed in Table 1. Samples for the environmental class are taken from CASR recordings [12]. The subclasses correspond the classes in CASR (car, restaurant, road). The drum samples are acoustic drum sequences used by Paulus and Virtanen [13]. The rest of the music class are from RWC Music Database [14], acoustic class is from RWC Jazz Music Database, electroacoustic is from RWC Popular Music Database, and Symphony is from RWC Classical Music Database. Sing mainclass, which contains only monophonic singing, was taken from Vox database presented in [15]. The speech samples are from the CMU Arctic speech database [16].

All the samples in our database are 10 seconds long. The length of speech samples in Arctic database are 2-4 seconds, thus the samples from each speaker are combined to result in 10-second samples. Original samples in the other databases are longer than 10 seconds, thus random 10 second clips are cut from those.

5.1. Evaluation procedure

One signal at the time is drawn from the database to serve as a query signal. This query signal is compared against the other signals in database in order to find near similar samples. This procedure is repeated for 10 random signals from each class. Altogether $10(n-1) * number\_of\_classes$ comparisons are performed, where n is the total number of signals in the database. K-NN search and ϵ-range query were tested. If the example and retrieved signal are labelled in the same class, the database signal is seen as correctly retrieved from the database.

Averages of recall and precision rates of classes are used to present the results. Recall reveals the portion of similar signals retrieved from the database:

Main class	Sub class
Environmental (231)	Inside car (151)
	In restaurant (42)
	Traffic (38)
Music (620)	Acoustic (264)
	Drums (56)
	Electroacoustic (249)
	Symphony (51)
Sing (165)	Humming (52)
	Singing (60)
	Whistling (53)
Speech (316)	Speaker1 (50)
	Speaker2 (47)
	Speaker3 (44)
	Speaker4 (40)
	Speaker5 (47)
	Speaker6 (38)
	Speaker7 (50)

Table 1: *Classes.*

$$recall(class) = \frac{N_{ccs}}{n_{class}(n_{class}-1)}, \tag{4}$$

where n_{class} is the number of samples in the class, and N_{ccs} means the number of correctly retrieved samples from this class.

Precision gives the portion of correctly retrieved samples from all the retrieved signals:

$$precision(class) = \frac{N_{ccs}}{N_D}, \tag{5}$$

where N_D is the total number of samples retrieved from certain class when the example signal is from this class.

5.2. Results

The results from compression based method using different audio coding algorithms compared to other methods in k-NN search when $k = 20$ are presented in Table 2. The results of ϵ-range query with different values of ϵ are illustrated in Figure 2.

The proposed method outperforms the reference methods in ϵ-range query with large values of ϵ. This means it is the most accurate method when the aim is to retrieve all the similar samples from the database. In k-NN search using k=20, the results were also relatively good but slightly lower than with the best feature-based method.

There were only minor differences between different audio codecs, AAC resulting in the best average results. Using no audio codec at all gave very poor results. This was expected considering that compression algorithms require an identical strings to compress and in wave format already a very small change generates different codewords. Similar effect can be seen when using higher bitrates in audio codecs thus the lower bitrates gave the best results.

6. CONCLUSIONS

In this paper, a novel approach to query by example for audio signals was presented. First, perceptually important characteristics of a signal are retained by using a perceptual audio coder. Then

Coding method	Prec. main %	Prec. sub %
No codec	29.0	8.7
MP3 8 kbit/s	94.1	68.8
AMR 8 kbit/s	96.5	83.0
AAC 10 kbit/s	96.5	85.5
Mahalanobis distance	97.3	92.6
Likelihood of GMMs	94.0	86.8
Histogram method	85.6	75.4
Euclidean distance of GMMs	97.5	95.7
KL distance of GMMs	97.5	90.8

Table 2: *Precision values for main classes and sub classes for different audio coding methods, and feature based methods with k-NN search, when k=20. Gzip is used as a compressor.*

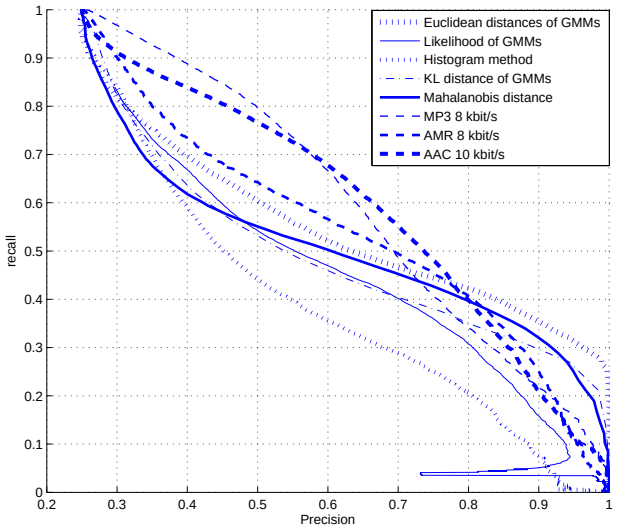


Figure 2: *ϵ-range results of different methods with different values of ϵ.*

coded audiofiles are compressed using standard lossless compression techniques and similarity is estimated from the compression ratios of individual files and combined files. The compression-based similarity metric does not require the setting of any parameters nor does it require any knowledge about the topic at hand.

The compression-based method was tested against the existing query by example methods. In ϵ-range query it outperformed the other methods at high recall rates and also in k-NN query it gave competitive results. This reveals that considering the simplicity of the proposed method, it is very practical for many applications. Especially ones, where there is very little knowledge about the contents of the database beforehand and thus, choosing the right features is impossible.

7. ACKNOWLEDGEMENTS

This work was supported by the Academy of Finland, project No. 213462 (Finnish Centre of Excellence program 2006 - 2011).

8. REFERENCES

[1] E. Pampalk, *Computational Models of Music Similarity and their Applications in Music Information Retrieval*, Ph.D. thesis, Technische Universitat, Wien, 2006.

[2] M. Mandel and D. Ellis, "Song-level features and support vector machines for music classification," in *Proc. 6th International Conference on Music Information Retrieval*, 2005.

[3] M. Helén and T. Lahti, "Query by example methods for audio signals," in *Proc. 7th IEEE Nordic Signal Processing Symposium*, Reykjavik, Iceland, June 2006, pp. 302–305.

[4] M. Helén and T. Virtanen, "Query by example methods of audio signals using Euclidean distance between Gaussian mixture models," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, USA, 2007.

[5] H. Ferhatosmanoglu, E. Tuncel, D. Agrawal, and A. El Abbadi, "Approximate nearest neighbor searching in multimedia databases," in *Proc. 17th IEEE International Conference on Data Engineering*, Heidelberg, Germany, Apr. 2001.

[6] K. Brandenburg, "Perceptual coding of high quality digital audio," in *Applications of Digital Signal Processing to Audio and Acoustics*. 1998.

[7] C. H. Bennett, P. Gács, M. Li, P. M. B. Vitányi, and W. H. Zurek, "Information distance," *IEEE Transactions on Information Theory*, vol. 44, no. 4, pp. 1407–1423, July 1998.

[8] C. Costa-Santos, J. Bernandes, P. M. B. Vitányi, and L. Antunes, "Clustering fetal heart rate tracings by compression," in *Proc. 19th IEEE International Symposium on Computer-Based Medical Systems*, Salt Lake City, Utah, USA, 2006.

[9] R. Cilibrasi and P. M. B. Vitányi, "Clustering by compression," *IEEE Transactions on Information Theory*, vol. 51, no. 4, pp. 1523–1545, Apr. 2005.

[10] E. Keogh, S. Lonardi, and C. Ratanamahatana, "Towards parameter-free data mining," in *Proc. 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Seattle, WA, USA, Aug. 2004.

[11] M. Li, X. Chen, X. Li, B. Ma, and P. M. B. Vitányi, "The similarity metric," *IEEE Transactions on Information Theory*, vol. 50, no. 12, pp. 3250–3264, 2004.

[12] V. Peltonen, J. Tuomi, A. Klapuri, J. Huopaniemi, and T. Sorsa, "Computational auditory scene recognition," in *Proc. 2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*, Florida, USA, May 2002.

[13] J. Paulus and T. Virtanen, "Drum transcription with non-negative spectrogram factorisation," in *Proc. 13th European Signal Processing Conference*, Antalya, Turkey, Sept. 2005.

[14] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, "RWC music database: Popular, classical, and jazz music databases," in *Proc. 3rd International Conference on Music Information Retrieval*, Oct. 2002.

[15] T. Viitaniemi, A. Klapuri, and A. Eronen, "A probabilistic model for the transcription of single-voice melodies," in *Proc. 2003 Finnish Signal Processing Symposium (FINSIG'03)*, Tampere, Finland, May 2003, pp. 59–63.

[16] J. Kominek and A. Black, "The cmu arctic speech databases," in *Proc. 5th ISCA Speech Synthesis Workshop*, Pittsburgh, USA, 2004, pp. 223–224.

THE REACTION SYSTEM: AUTOMATIC SOUND SEGMENTATION AND WORD SPOTTING FOR VERBAL REACTION TESTS

Gunnar Eisenberg, Thomas Sikora

Communication Systems Group
Technical University of Berlin, Germany
{eisenberg|sikora}@nue.tu-berlin.de

ABSTRACT

Reaction tests are typical tests from the field of psychological research and communication science in which a test person is presented some stimulus like a photo, a sound, or written words. The individual has to evaluate the stimulus as fast as possible in a predefined manner and has to react by presenting the result of the evaluation. This could be by pushing a button in simple reaction tests or by saying an answer in verbal reaction tests. The reaction time between the onset of the stimulus and the onset of the response can be used as a degree of difficulty for performing the given evaluation.

Compared to simple reaction tests verbal reaction tests are very powerful since the individual can simply say the answer which is the most natural way of answering. The drawback for verbal reaction tests is that today the reaction times still have to be determined manually. This means that a person has to listen through all audio recordings taken during test sessions and mark stimuli times and word beginnings one by one which is very time consuming and people-intensive.

To replace the manual evaluation of reaction tests this article presents the REACTION (*Reac*tion *T*ime Determinati*on*) system which can automatically determine the reaction times of a test session by analyzing the audio recording of the session. The system automatically detects the onsets of stimuli as well as the onsets of answers. The recording is furthermore segmented into parts each containing one stimulus and the following reaction which further facilitates the transcription of the spoken words for a semantic evaluation.

1. INTRODUCTION

There are three main classes of reaction tests, the plain *Reaction Time Test*, the *Stroop Test*, and the *Association Test*, which investigate different psychological phenomena. The typical setup for each of them is that a test person watches a screen on which a visual stimulus is presented. To gain the attention of the participant the stimulus is presented together with an alerting sound like a beep. After evaluating the stimulus the participant reacts by saying his answer. Simple java examples of non-verbal reaction tests can be found on the web [1, 2, 3].

In plain *Reaction Time Tests* the participant does not have to make any decisions about the presented stimulus [1]. He just has to acknowledge the perception of the stimulus as fast as possible. The test simply evaluates the reaction time's length. An example of a Plain Reaction Time Test could be that a red dot appears somewhere on the screen at random intervals in time. The participant has to say the word "dot" every time he discovers it.

Stroop Tests, named after their inventor, try to create some interference in the test person's consciousness between trained actions and cognitive abilities [2, 4]. Therefore the participant has to make a decision about the stimulus which is interfered by some opposing property of the stimulus itself. A well known example is reading color names (e.g. red, green, blue, etc.) which are printed in a different color or vice versa. Another example is naming the highest number out of a set of printed numbers with the smaller numbers being printed in a much bigger font than the higher numbers.

In *Association Tests* the test person is presented a picture or a word, often a noun (e.g. love, death, pleasure, etc.) on which he has to answer a certain emotional association (e.g. good, bad, embarrassing etc.) [3, 5].

Reaction test sessions are usually recorded on audio or videotape to be evaluated afterwards. On the audio track of the recordings the audible alert signals marking new stimuli and the answers of the participants are recorded. In this simple setup no additional information like time stamps or electronic markers for the onsets of new stimuli is recorded. This means that the recording is the only resulting material from the test session.

The advantages of using this simple setup is that it is very portable and investigators only have to take a minimum care of technical issues. The playback device usually is a laptop or sometimes a video cassette recorder with a TV-screen. The recording device is often an analogue dictating machine placed somewhere near the test person. Since until now the tests are manually evaluated afterwards, the poor recording quality is not impairing the evaluation as long as all answers can be understood.

To replace the manual evaluation an automatic evaluation system, like the one presented in this article, processes the sessions' recordings as input. It has to detect the recorded alert signals to determine the stimuli onsets and the onsets of the recorded answers. The system has to deal with the recording's poor quality like a high ground noise level, crackles, bad leveling and clipping.

2. PREVIOUS APPROACHES

The automatic measurement of reaction times in reaction tests breaks down into two tasks. One task is detecting the alert sounds' onsets marking the beginning of new stimuli. These onsets are the borders of segments, each containing a new stimulus and an answer. The other task is finding the onsets of the answered words. Both tasks have to be performed under noisy conditions.

Although there is actually no system which approaches the automatic evaluation of reaction tests directly there are approaches which perform tasks similar to the two subtasks mentioned above.

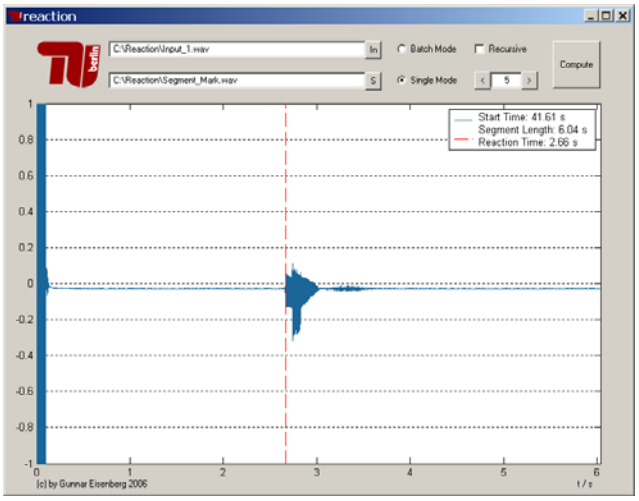


Figure 1: *REACTION's graphical user interface showing a computed segment together with the word's onset.*

Matsunaga et al. have presented a procedure for automatically segmenting broadcast news into speech, music and jingles (comparable to the given alert signal) and other classes [6]. In a noise free environment the detection rate for speech is 95.0 % and the detection rate for jingles is 87.7 %. The system has not been tested under noisy conditions.

Kim and Sikora have compared different algorithms for automatically segmenting sounds from different speakers in broadcast audio material [7]. The system does not need a priori information about the number of speakers and its recognition rate is 93.2 % for a scenario comparable to the scenario given in this work but with clean speech. Although the presented algorithms work well with clean speech the recognition rate drops with noisy environments.

Dufaux et al. have presented a system for automatic sound detection for noisy environments [8]. It detects impulsive sounds and is used for surveillance purposes. Their system has a recognition rate of up to 85.1 % for a SNR of 10 dB. The system could be useful for finding the alerts marking new segments but for an applicable system the recognition rate is still not high enough.

The work of Spina and Zue on automatic segmentation of general audio data [9] focuses on the training of segmentation systems which operate on noisy environments. Their work also shows the difficulty of trained recognition systems to deal with noise at all.

Various methods have been proposed for general onset detection which can also help solving the problem [10]. The recognition rates for onsets in a comparable scenario range from 70 % to 90 % and the problem of distinguishing between stimuli onsets and word onsets in an error prone environment would remain.

The cited approaches are developed to meet the requirements of a general case scenario. Therefore they turned out not to be robust enough to be directly applicable. As a result the REACTION system uses a different signal processing approach custom made for the given reaction test scenario.

3. THE REACTION SYSTEM

The user interface of the REACTION system can be seen in figure 1. The system needs two wave files in pcm-coded format,

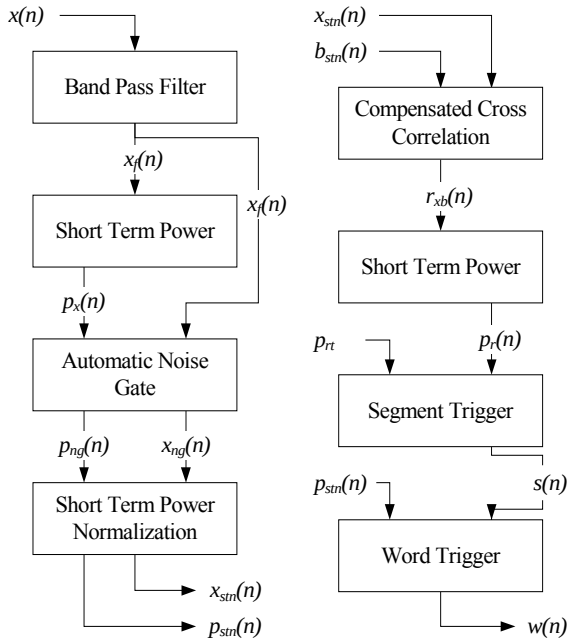


Figure 2: *Flowchart of the REACTION system showing the pre-processing chain (left) and the main processing chain (right). Input signals are shifted to the left, output signals are shifted to the right.*

mono or stereo with a minimum sample rate of 8000 Hz as input files for processing. One is the session's recording and the other is the short alert signal that marks the onsets of stimuli.
The process which is performed by REACTION is segmenting the session's recording by searching for the given alert signal so that each segment starts with the onset of a new stimulus. Further the onset of the test person's response is detected and the reaction time i.e. the time between the segment's start and the word's onset is determined. REACTION can operate on single sessions' recordings or in batch mode on several recorded sessions in one or more folders. The distinction between single or batch mode is done with the radio buttons in the upper right part of the interface. In batch mode the user can also set the system to crawl the selected folder recursively by checking the field "Recursive".
The program together with a manual and examples can be downloaded at our institute's website [11]. The usage is free for research purposes and in non commercial applications.

4. ALGORITHM

The two input signals of the system are $x(n)$ which is the session's recording and $b(n)$ which is the alert signal that marks the onsets of stimuli with n denoting the sample index. The algorithm's flowchart is shown in figure 2. It is divided in a pre-processing stage which operates on $x(n)$ and $b(n)$ and the main process. A part of a typical session's recording can be seen in figure 3.

4.1. Pre-Processing

The session's recording $x(n)$ is first band pass filtered with a second order Butterworth filter with cutoff frequencies at 50 Hz and 3900 Hz. This eliminates high frequency glitches and DC-

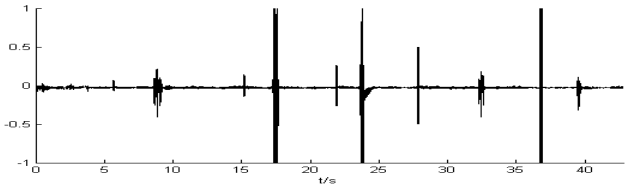


Figure 3: *Part of a typical input signal x(n). The ground noise level together with the bursts being alert sounds or spoken words can clearly be seen.*

offsets together with other low frequent rumble. After this initial filtering the signal is resampled to $f_s = 8000$ Hz for further processing.

For the resulting signal $x_f(n)$ the short term power $p_x(n)$ is computed with a window size of 25 ms, resulting in $N = 200$ for $f_s = 8000$ Hz:

$$p_x(n) = \frac{1}{N} \sum_{k=-N/2}^{N/2-1} {x_f}^2(n+k) \; . \tag{1}$$

To eliminate ground noise an automatic noise gate is applied to the signal. Because of the nature of $x(n)$ most of its samples will neither contain speech nor parts of the alert but only the ground noise. Therefore a histogram is build to count the occurrences of the different values of $p_x(n)$. The value of of $p_x(n)$ which occurs most often will represent the ground noise level p_{gn}. All samples of $x_f(n)$ and $p_x(n)$ will be set to zero if their level is smaller than $2 \cdot p_{gn}$ resulting in the signals $x_{ng}(n)$ and $p_{ng}(n)$:

$$p_{ng}(n) = \begin{cases} 0 & \left| p_x(n) < 2\,p_{gn} \right. \\ p_x(n) & \left| p_x(n) \geq 2\,p_{gn} \right. \end{cases} , \tag{2}$$

$$x_{ng}(n) = \begin{cases} 0 & \left| p_x(n) < 2\,p_{gn} \right. \\ x_f(n) & \left| p_x(n) \geq 2\,p_{gn} \right. \end{cases} . \tag{3}$$

The signal is further normalized by a modified version of the short term power. Therefore the one sided decay envelope $v^*_{png}(n)$ of $p_{ng}(n)$ is computed:

$$v^*_{png}(n) = \max\left(p_{ng}(n), \quad v^*_{png}(n-1) \cdot \Delta \right). \tag{4}$$

The half value time for the exponential decay envelope is set to 220 ms, resulting in $\Delta = 99.961\%$ for $f_s = 8000$ Hz. The two sided decay envelope $v_{png}(n)$ is gained by applying equation (4) again to the reversed signal of $v^*_{png}(-n)$. The output signal $x_{stn}(n)$ and its power $p_{stn}(n)$ can be obtained by normalizing $x_{ng}(n)$ and $p_{ng}(n)$ to the power envelope as given by the following equations:

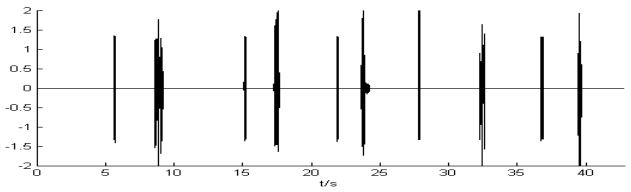


Figure 4: *The input signal after being pre-processed. The noise is gone and all bursts are normalized. The alert signals and spoken words can already be visually distinguished. The alert signals appear as cubic bursts whereas the words have a frayed shape.*

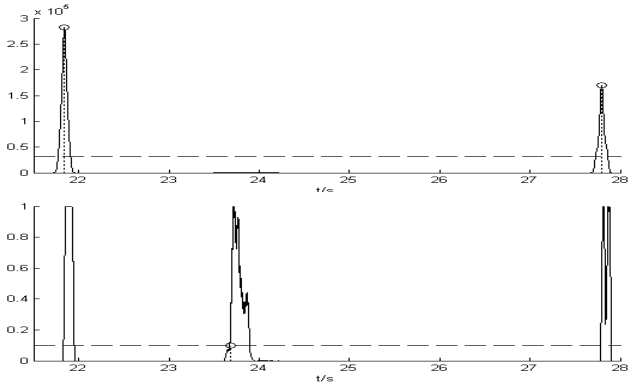


Figure 5: *The short term power of the cross correlation (top) and the short term power of the pre-processed input signal (bottom) together with horizontal dashed lines marking the static thresholds for new segments and word's onsets. The detected segment borders and word onsets are marked with vertical dotted pins.*

$$p_{stn}(n) = p_{ng}(n) / v_{png}(n) \; , \tag{5}$$

$$x_{stn}(n) = x_{ng}(n) / \sqrt{v_{png}(n)} \; . \tag{6}$$

The pre-processing of the alert signal $b(n)$ to form the signal $b_{stn}(n)$ is formed accordingly to the steps described above. Only the automatic noise gate can be omitted because the nature of the signal is that it has no silent passages.

After having passed the pre-process stage the signals $x_{stn}(n)$ and $b_{stn}(n)$ are band limited, they are eventually noise gated and normalized in a way that their short term power is unity for the alert passages as well as for the spoken words. Figure 4 shows the signal from the example used in figure 4 after being pre-processed.

4.2. Main Processing

The second processing stage is the main process in which the alert signal's onsets and the words' onsets are determined. Since the signals have fixed properties after pre-processing this determination can be computed in a straight forward process.

First $x_{stn}(n)$ and $b_{stn}(n)$ are cross correlated to build the correlation signal $r_{xb}(n)$. To get rid of the typical phenomenon of oscillation of the correlation signal the short term power $p_r(n)$ of $r_{xb}(n)$ is computed according to equation (1), again using a window size of 25 ms.

Since $x_{stn}(n)$ and $b_{stn}(n)$ both are normalized in terms of their short term power no dynamic leveling needs to be applied for using the correlation's short term power $p_r(n)$ as a trigger to get the segment's onsets. It can directly be compared to a static threshold p_{rt}. This threshold is automatically determined to be 15 % of the maximum short term power value $p_{bb}(n)$ of the autocorrelation $r_{bb}(n)$ from $b_{stn}(n)$. Every local maximum of $p_r(n)$ marks a new segment as given by the following equations if it lies in a set of taps M_j whose according values of $p_r(n)$ lie above that threshold:

$$s_j(n) = \begin{cases} 1 & \left| n = \arg\max_{n \in M_j} p_r(n) \right. \\ 0 & \left| \textit{otherwise} \right. \end{cases} , \tag{7}$$

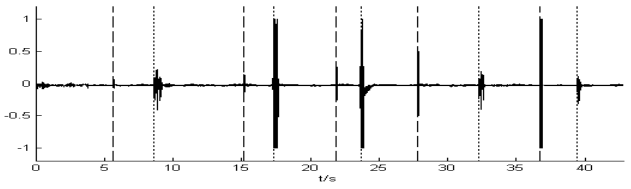


Figure 6: *The original input signal together with the segment borders (dashed lines) and the word's onsets (dotted lines).*

$$s(n) = \sum_j s_j(n). \tag{8}$$

The signal $s(n)$ which is also an output signal of the whole process, has the character of a trigger signal. It is 1 at the beginning of new segments and 0 elsewhere. To avoid multiple triggering the threshold has to be crossed for at least 10 ms (80 taps for f_s = 8000 Hz) which avoids triggering by glitches. Furthermore a new segment is only indicated if the last one is at least 50 ms gone (400 taps for f_s = 8000 Hz).

To find the word's onsets a slope technique is used. For every segment the word's onset is defined to be the first point in time where the normalized short term power $p_{stn}(n)$ of the signal $x_{stn}(n)$ reaches 25 % of its maximum, which is exactly 0.25 because of the normalization. To avoid triggering by glitches the threshold has to be crossed for at least 10 ms (80 taps for f_s = 8000 Hz). The signal $w(n)$ is derived from the words' onset times. It is 1 at the words' onsets and 0 elsewhere. Figure 5 shows parts of the signals $p_r(n)$ and $p_{stn}(n)$ for the example from figure 3 together with the generated triggers for segments and words' onsets. The resulting segmentation for the example signal can be seen in figure 6.

5. EVALUATION

The system was evaluated with real recordings of a reaction test. In this test 240 persons had to respond to 89 stimuli resulting in 21360 stimuli to be processed. The mean length of each test session's recording was 11.03 seconds and the total length of all evaluated recordings was 44:12 hours. The average reaction time determined in the tests was 3.30 seconds. The recordings were taken with an analogue dictating machine.

Although the quality of the recordings was quite poor, including the earlier mentioned flaws, the performance of the system was very good as it is depicted in figure 7. From the 21360 processed stimuli the REACTION system could segment 21162 segments (99.1 %) correctly. It has turned out that the system has never detected a new segment at a wrong point. Either the segment's border is detected correctly or it is missed completely. This behavior helps finding falsely segmented stimuli since they double the value of the determined (false) reaction time for the preceding segment. This marks these falsely segmented stimuli clearly as outliers in subsequent evaluations. Furthermore this behavior matches with outliers produced by semantic errors, i.e. when a person for some reason takes very long to respond to the presented stimulus. Therefore errors resulting from false segmentation can be ruled out quite easily afterwards.

From the 21162 correctly detected segments for 20583 words (97.3 %) the onset was detected correctly with an allowed tolerance of 15 ms. Compared to typical reaction times which are several seconds (in this case 3.30 s) the given tolerance is quite small. In total the number of correctly detected reaction times was 20583 (96.4 %).

Segmentation Rate	99.1 %
Onset Detection Rate	97.3 %
Reaction Time Detection Rate	96.4 %

Figure 7: *REACTION's Detection Rates*

6. CONCLUSIONS

The presented REACTION system can automatically detect reaction times from audio recordings of verbal reaction tests. It is indifferent against noise and other signal errors and because of its high recognition rate it is directly applicable and robust in everyday use.

7. REFERENCES

[1] G. Bradshaw, "Simple Choice Reaction Time," Available at http://epsych.msstate.edu/deliberate/SimpleRT/5.html, Accessed April 30, 2007

[2] E.Z. Yang, "Stroop Effect - Interactive Test," Available at http://www.thewritingpot.com/stroop/, Accessed April 30, 2007.

[3] T. Flynn, "Personality Test – Word Association Test," Available at http://www.similarminds.com/word/, Accessed April 30, 2007.

[4] C.M. MacLeod, P.A. MacDonald, "Interdimensional interference in the Stroop effect: uncovering the cognitive and neural anatomy of attention," *Trends in Cognitive Sciences*, vol. 4, no. 10, October 2000.

[5] G.R. Marshall, C.N. Cofer, "Associative Indices as Measures of Word Relatedness: A summary and Comparison of Ten Methods," *Journal of Verbal Learning and Verbal Behavior*, January 1964.

[6] S. Matsunaga, O. Mizuno, K. Ohtsuki, Y. Hayashi, "Audio Source Segmentation Using Spectral Correlation Features for Automatic Indexing of Broadcast News," in *Proc. 12th European Signal Processing Conference (EUSIPCO-2004),* Vienna, Austria, Sep. 2004.

[7] H.-G. Kim, T. Sikora, "Automatic segmentation of speakers in broadcast audio material," in *Proc. of SPIE*, Volume 5307, Storage and Retrieval Methods and Applications for Multimedia 2004, Dec. 2003.

[8] A. Dufaux, L. Besacier, M. Ansorge, F. Pellandini, "Automatic Sound Detection and Recognition for Noisy Environment" in *Proc. Xth European Signal Processing Conference (EUSIPCO-2000), Tampere,* Finland, Sep. 2000.

[9] M.S. Spina, V.W. Zue, "Automatic Transcription Of General Audio Data: Effect Of Environment Segmentation On Phonetic Recognition," in *Proc. 5th European Conference on Speech Communication and Technology (EUROSPEECH '97),* Rhodes, Greece, Sep. 1997.

[10] S. Dixon, "Onset Detection Revisited," in *Proc. Workshop on Digital Audio Effects (DAFx'06),* Montreal, Canada, Sep. 2006.

[11] G.Eisenberg, "REACTION – Reaction Time Determination Software" Available at http://www.nue.tu-berlin.de/wer/eisenberg/reaction/, Accessed June 21, 2007.

THE BEATING EQUALIZER AND ITS APPLICATION TO THE SYNTHESIS AND MODIFICATION OF PIANO TONES

Jukka Rauhala

Laboratory of Acoustics and Audio Signal Processing
Helsinki University of Technology, Espoo, Finland
jukka.rauhala@acoustics.hut.fi

ABSTRACT

This paper presents an improved method for simulating and modifying the beating effect in piano tones. The beating effect is an audible phenomenon, which is characteristic to the piano, and, hence, it should be accounted for in realistic piano synthesis. The proposed method, which is independent of the synthesis technique, contains a cascade of second-order equalizing filters, where each filter produces the beating effect for a single partial by modulating the peak gain. Moreover, the method offers a way to control the beating frequency and the beating depth, and it can be used to modify the beating envelope in existing tones. The results show that the proposed method is able to simulate the desired beating effect.

1. INTRODUCTION

The beating effect is one of the audible characteristics in piano tones [1]. It occurs due to the coupling of detuned strings. Even if there is only one string per a key, as in the first keys of the piano, beating can be present due to false coupling [2]. As the beating effect is a perceptually important phenomenon, it must be taken into account in a realistic piano synthesis model.

Various beating effect simulations have been proposed for digital waveguide synthesis. In the first waveguide models the beating effect was produced with parallel detuned string models [3, 4]. Bank suggested a resonator-based approach, where a resonator is tuned close to the frequency of the target partial, which produces the beating effect due to frequency modulation [5, 6]. In addition, a multi-rate version of the resonator-based approach has been proposed [7]. In the resonator-based approach, the frequency of the partial must be known in order to control the beating frequency. Moreover, the approach does not provide straight-forward control over the depth of beating. Additionally, Bank and Sujbert have suggested a method using pitch-shift to produce the beating effect [8].

Rauhala et al. [9] proposed a beating model, where the beating effect is, first, simulated by separating the partial from the signal with a bandpass filter. Then, the partial is modulated with a low-frequency oscillator (LFO). Finally, the modulated partial is added to the original signal. This approach does not require exact knowledge of the frequency of the partial and it provides an easy way to control the depth of the beating. On the other hand, the depth control is not very accurate due to the mixing of signals (however, Järveläinen and Karjalainen [10] suggest that the perception of the depth of the beating is quite poor), and the mixing can produce some uncontrollable features in the produced sound.

In this paper, an improved beating effect method is proposed based on [9]. The main idea in this method is to produce the mod-

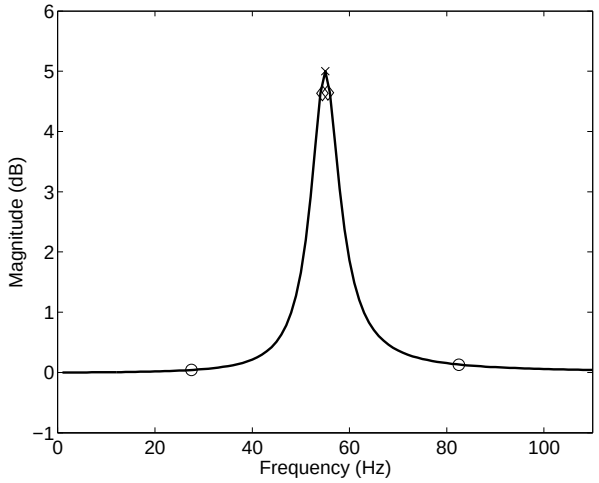


Figure 1: *An example of the magnitude response of the equalizing filter (f_c=55.0 Hz, f_{bw}=5.5 Hz, f_s=44100 Hz, K=5.0 dB) used in the proposed method. This demonstrates the case where the second partial is modified with the method (f_0=27.5 Hz). The cross denotes the magnitude response at the target partial frequency (5.0 dB), while the circles denote the response at the adjacent partial frequencies (0.04 dB for the first partial and 0.13 dB for the third partial). The filter's magnitude in the case where the estimated partial frequency is biased by ± 1 % is denoted with diamonds (4.63 dB for -1.0 % bias and 4.64 dB for +1.0 % bias).*

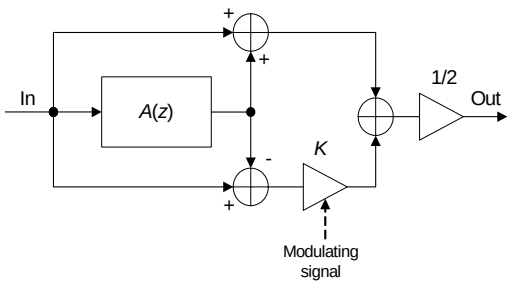


Figure 2: *Block diagram of the equalizing filter [11].*

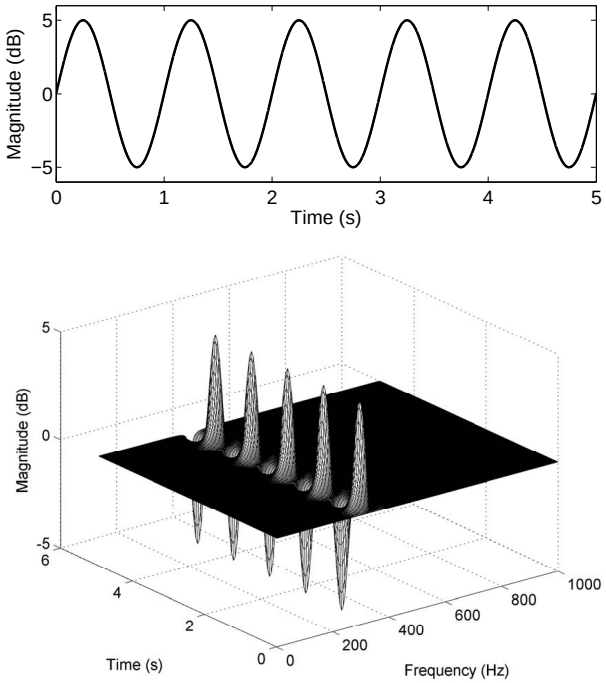


Figure 3: *(Bottom) An example of the magnitude response of the equalizing filter (f_{c}=327.0 Hz, f_{bw}=65.4 Hz, f_{s}=44100 Hz) in time, when peak gain K is modulated (G_{b} = 5 dB and beating frequency 1.0 Hz). (Top) The corresponding modulation signal envelope.*

ulation with the equalizing filter by controlling its peak gain. This results in a simpler structure than in the previous method. Moreover, it offers accurate control over the beating frequency and the beating depth. Additionally, it can be generalized to produce any kinds of envelopes for certain partials in an arbitrary audio signal. Also, the simulation process is accurately controlled since there is no need to mix signals as in the previous method. In addition, the method can be used for modifying and even cancelling the beating effect of certain partials in existing tones.

This paper is organized as follows. The proposed method is introduced in Section 2. The results from applying it for simulating the beating effect in synthetic tones and for modifying the partial envelopes of recorded tones are then presented. Finally, the conclusions are shown in Section 4.

2. PROPOSED METHOD

A second-order equalizing filter, proposed originally by Regalia and Mitra [11], was chosen to produce the beating effect in the proposed method, because it provides control over the peak gain via a single parameter. Moreover, the magnitude response of the filter is suitable for modifying a single partial, as it can have a narrow peak at the desired frequency and a flat response elsewhere. The transfer function of the equalizing filter can be written as [11, 12]

$$H_{\mathrm{EQ}}(z) = \frac{1}{2}(1+K) + \frac{1}{2}(1-K)A(z), \qquad (1)$$

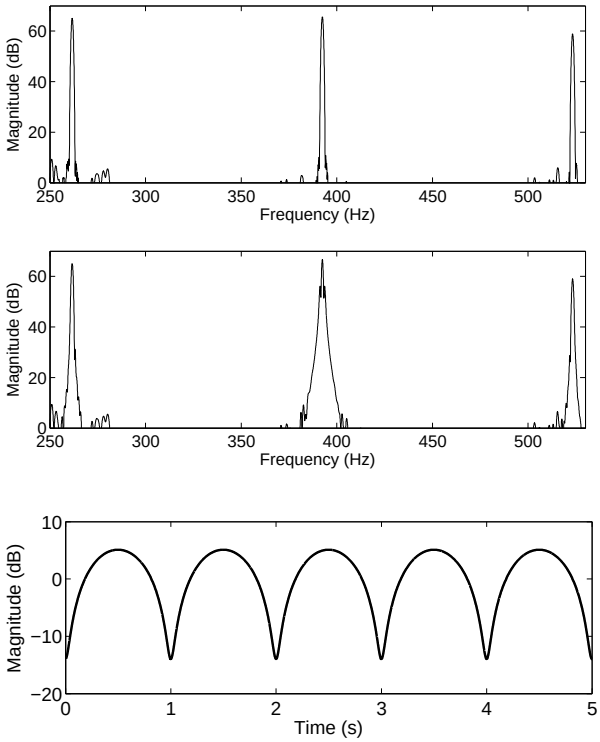


Figure 4: *Magnitude responses of the original signal (top) and the processed signal (middle), which has been modulated with the equalizing filter (f_{c}=392.4 Hz, f_{bw}=26.2 Hz, f_{s}=44100 Hz, G_{b} = 5 dB). The corresponding modulation envelope (bottom) has been obtained by examining the resulting envelope from a frequency modulated signal containing two sinusoidal components. The original signal is a synthetic piano tone (key C_2, f_0=130.8 Hz) produced with the waveguide piano synthesis model [9].*

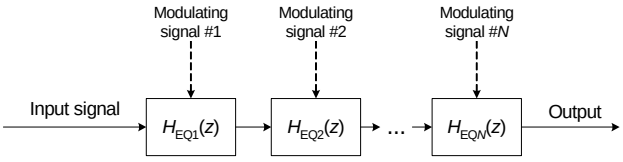


Figure 5: *Block diagram of the general structure for modifying partial envelopes.*

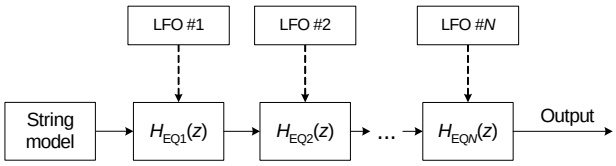


Figure 6: *Block diagram of the proposed method applied to beating effect simulation in piano tones.*

where

$$A(z) = \frac{a - \cos(\frac{2\pi f_c}{f_s})(1+a)z^{-1} + z^{-2}}{1 - \cos(\frac{2\pi f_c}{f_s})(1+a)z^{-1} + az^{-2}}, \quad (2)$$

$$a = \frac{1 - \tan(\frac{\pi f_{bw}}{f_s})}{1 + \tan(\frac{\pi f_{bw}}{f_s})}, \quad (3)$$

f_c is the center frequency of the peak, f_{bw} is the peak bandwidth, f_s is the sampling frequency, and K is the peak gain. In this work, f_{bw} is determined as $0.2f_0$, where f_0 is the fundamental frequency.

Figure 1 shows an example of the filter's magnitude response. Since the filter's effect on the adjacent partials is minimal (around 0.1 dB in this case), it suggests that the filter does not produce audible effects on the adjacent partials. Moreover, the filter is robust against inaccurate partial frequency estimations, as a 1.0 % bias leads to a peak magnitude of 4.6 dB instead of 5.0 dB in this case.

The filter can be structured such that K is a single independent multiplier as seen in Figure 2. Zölzer [12] showed that the magnitude response of this filter is slightly asymmetric, which can be fixed by modifying a to be dependent on K if $K < 1$. However, the asymmetric property of the magnitude response is not audible as the bandwidth of the peak is very narrow in this case. Hence, we propose to use Eq. (2) as such in this method.

In this method, K is modulated with a control signal. For instance, in case of the beating effect, K can be determined as

$$K(n) = 10^{\frac{G_b y_{LFO}(n)}{20}}, \quad (4)$$

where n is time in samples, G_b is the desired beating depth in dB, and y_{LFO} is the signal produced with the LFO generator. An example of the resulting magnitude response of the filter, when K is modulated with the LFO, is shown in Figure 3.

It is important to take into account that by modulating filter coefficient K the filter becomes time-variant. When the modulation signal resembles an envelope, which can be produced with frequency modulation, the only major effects on the spectrum of the resulting modulated tone are the two sidelobes that cause the beating effect, as seen in Figure 4. Moreover, there will be no transient effects [13, 14], since the structure does not have a feedback loop after coefficient K.

In order to produce the effect for multiple partials, a cascade of equalizing filters can be used. The generalized method is shown in Figure 5, and the method applied for simulation of beating effect for several harmonics is presented in Figure 6.

3. APPLICATION EXAMPLES AND RESULTS

In this section, the results from applying the proposed method for simulation of the beating effect for synthetic tones are presented. It is then shown how the method can be used for modifying partial envelopes in recorded tones.

3.1. Simulation of the beating effect for synthetic tones

The proposed method was incorporated into the previously presented waveguide piano model [9]. The piano string model includes a dispersion filter [15], a loss filter [16], a delay line, and a fractional delay filter [17] for tuning the fundamental frequency. In addition, the string model is excited with a parametric excitation

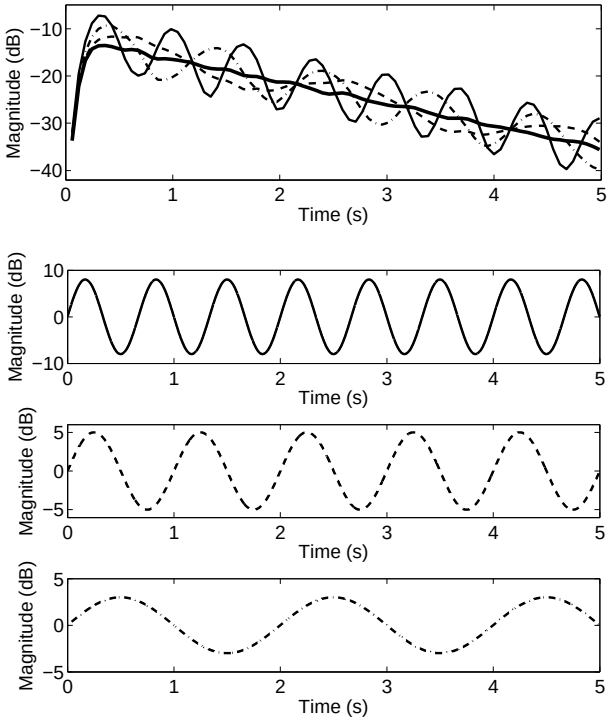


Figure 7: *(Top) The envelope of the 5th partial produced by the piano synthesis model without the beating method (thick line), and with the proposed beating method at various parameter values: G_b = 8 dB and beating frequency 1.5 Hz (solid line), G_b = 5 dB and beating frequency 1.0 Hz (dashed line), and G_b = 3 dB and beating frequency 0.5 Hz (dash-dotted line). The bottom three panes show the corresponding modulation signal envelopes. The fundamental frequency is 65.4 Hz (key C_3) and the inharmonicity coefficient value is* 1.5×10^{-4}.

method [18]. In the first test, the beating effect was added to a single partial with different beating frequencies and beating depths. The results, which are shown in Figure 7, suggest that the method is able to produce the beating effect accurately at various beating frequencies and beating depths.

Also, the robustness of the simulation method was evaluated by using inaccurate partial frequencies in the simulation biased by 1 %, 2 %, and 5 %. In sound synthesis, partial frequencies can be estimated accurately if the phase delay response of the dispersion filter can be calculated. However, if the dispersion filter is controlled in real-time [15], the partial frequency estimations might be slightly inaccurate. For example, frequency modulation-based methods are not robust against inaccurate partial frequency estimations, because a bias in the estimation will significantly affect the beating frequency and the depth of the beating effect. Figure 8 shows that the frequency of the beating effect remains the same in all cases, whereas the depth of the effect decreases with large bias values. The beating effect is difficult to detect when the bias is 5 %, but at 2 % it can be seen clearly in Figure 8. However, the estimation error is usually below 1 % within the bandwidth where the dispersion phenomenon is perceived [15, 19]. Hence, the proposed simulation method is suitable for sound synthesis.

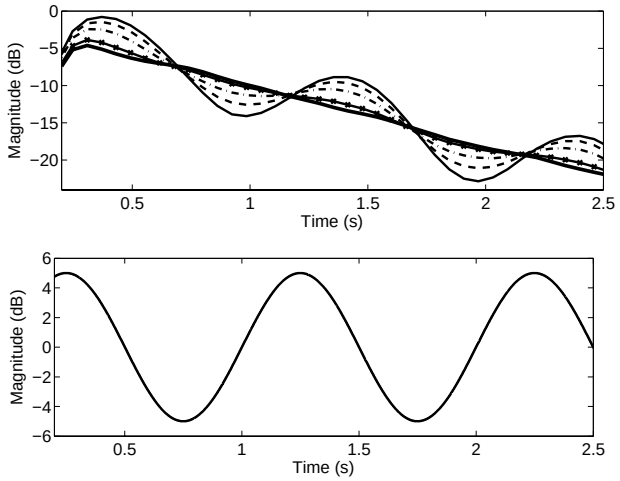


Figure 8: *(Top) The envelope of the 5th partial produced by the piano synthesis model including the beating method, when the partial frequency is biased by 0 % (solid line), 1 % (dashed line), 2 % (dash-dotted line), and 5 % (line with crosses). The envelope produced without the beating model is denoted as thick line. The fundamental frequency is 65.4 Hz (key* C_2*), and the inharmonicity coefficient value is* 1.5×10^{-4}*. (Bottom) The modulation signal envelope.*

Next, the method was used to simulate a realistic case, where the beating effect is present in the envelopes of multiple partials of the synthetic piano tone. Two synthetic tones were produced, where the first one used modulation signals obtained from the measured partial envelopes, and the latter used a full-wave rectified sinusoidal LFO (in real-time sound synthesis, the latter is better as there is no need to store large modulation signals for individual partials). Figure 9 displays the results, which show that the method is able to simulate the desired beating effect. The tone, which was produced using the measured partial envelopes shown in Figure 10, has very similar partial envelopes compared to the target tone. The latter tone with rectified sinusoidal modulation (modulation signals are shown in Figure 11) captures the dominating trends in partial envelopes, which might be enough for real-time sound synthesis, as the partial envelopes cannot be perceived very accurately [10].

3.2. Modification of the partial envelopes in recorded tones

The proposed method is not only able to simulate the beating effect for waveguide synthesis, but it can morph the partial envelopes in audio signals with various kinds of modulating signals. In order to demonstrate this, the method was used for modifying partial envelopes in a recorded piano tone in two ways. First, the beating effect of a single partial was increased. Figure 12 shows the original signal and the modified signal, where the beating effect of the second partial has been increased without affecting other partial envelopes.

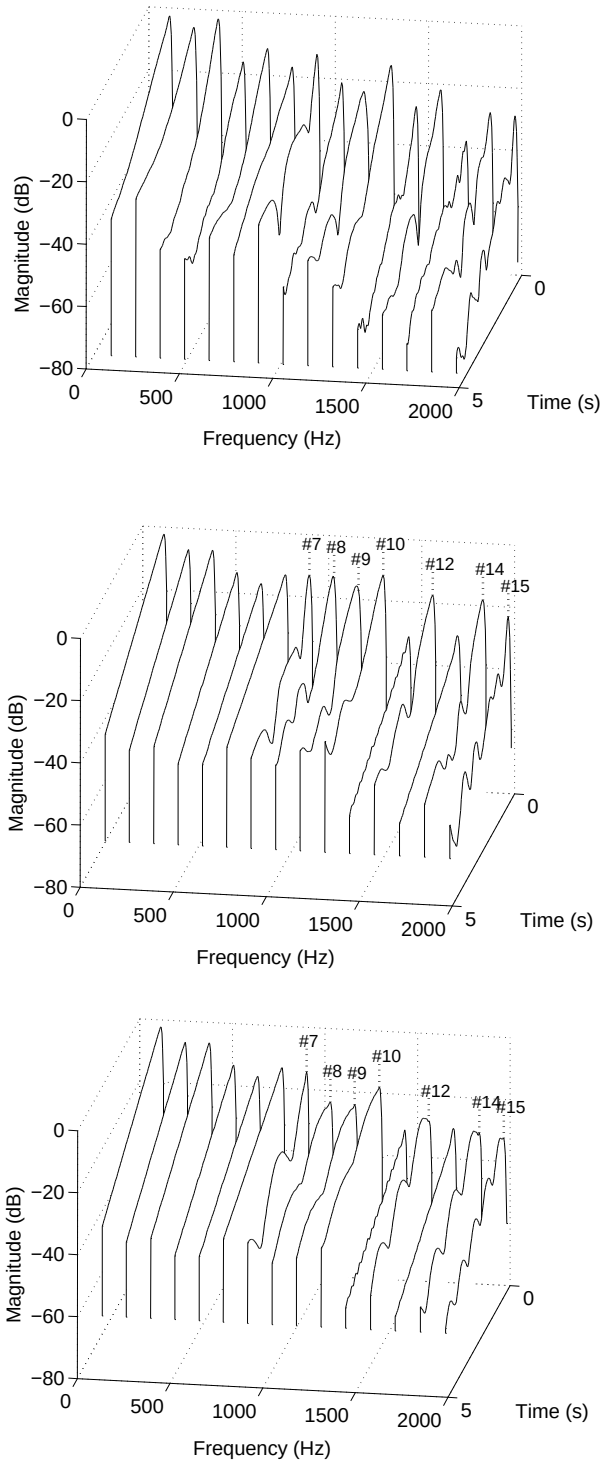


Figure 9: *(Top) Partial envelopes extracted with the short-time Fourier transform (STFT) from a recorded piano tone (*f_0*=129.1 Hz, key* C_3*), (middle) a synthetic tone produced with the proposed method using exact partial envelopes obtained from the recorded tone as modulation signals (the envelopes are shown in Figure 10), and (bottom) a synthetic tone produced with the proposed method using rectified sinusoidal modulation approximating the partial envelopes (the modulation signal envelopes are shown in Figure*

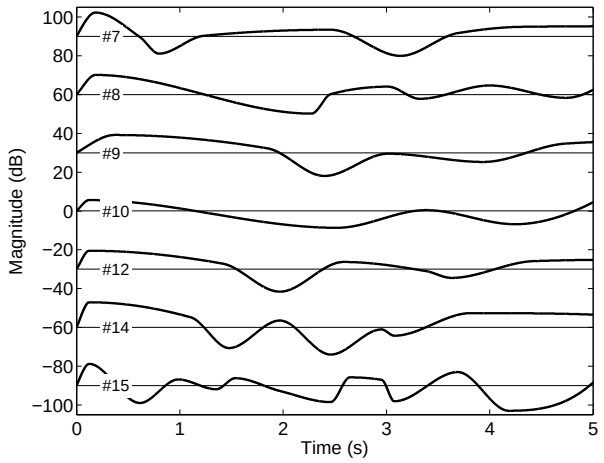


Figure 10: *The modulation envelopes used in the middle figure of Figure 9.*

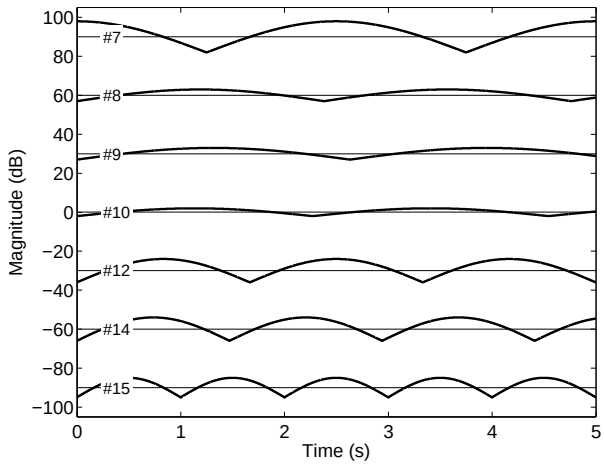
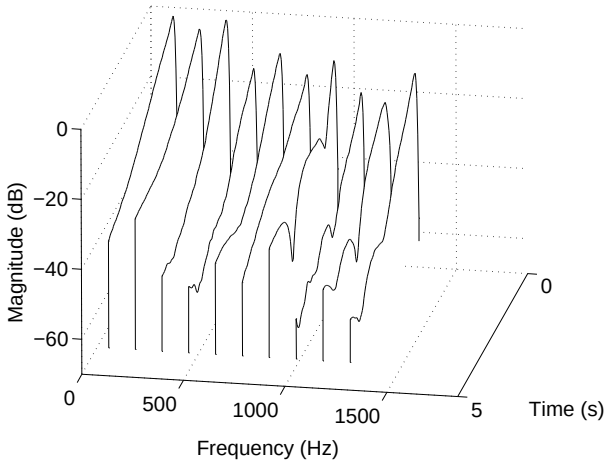


Figure 11: *The modulation envelopes used in the bottom figure of Figure 9.*

Next, the beating effect of first ten partials was cancelled. The modulation signals shown in Figure 13 were obtained by, first, eliminating the general decay rate in the determined partial envelopes and then inverting the resulting envelopes. The partial frequencies were determined manually in these examples. It can be seen in Figure 14, which shows the original signal and the modified signal, that the beating effect has been reduced significantly except for one dip in the envelope of the seventh partial. The reason for this dip is that the magnitude of the notch in the original envelope is larger than 20 dB, which is more than what the beating equalizer is capable of amplifying without causing undesired effects on the tone. Hence, the dip in the modulation signal had to be smoothed in order to prevent undesired effects. Sound examples are available in the web [1].

Modification of recorded tones is an exciting feature, which

[1] http://www.acoustics.hut.fi/publications/papers/dafx07-beq/



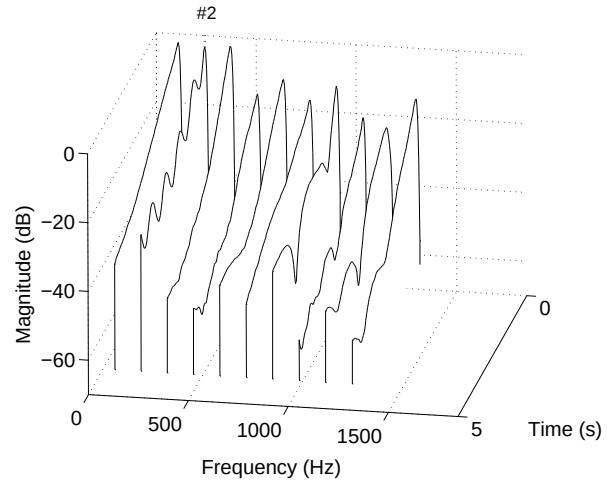


Figure 12: *(Top) The partial envelopes extracted with STFT from the original recorded piano tone (f_0=129.1 Hz, key C_3), and (bottom) the partial envelopes of a modified tone, where the second partial envelope has been modulated with the LFO with parameter values G_b = 5 dB and beating frequency = 1 Hz.*

can be used for sound analysis purposes. For instance, it can be used for minimizing the effect of beating when calibrating sound synthesis models. Then, the beating effect simulation can be calibrated separately. Secondly, it can be used for synthesizing tones for experiments evaluating the perception of the beating effect [10] by modifying recorded tones and controlling the beating effect.

4. CONCLUSIONS

This paper proposes an improved beating-effect simulation by modulating the peak gain of an equalizing filter. The proposed method is simple and it offers accurate control over the beating frequency and the beating depth in a straight-forward manner, as seen in the test results. Moreover, it is unnecessary to know the exact frequency of the partial, as in the resonator-based approach, since the

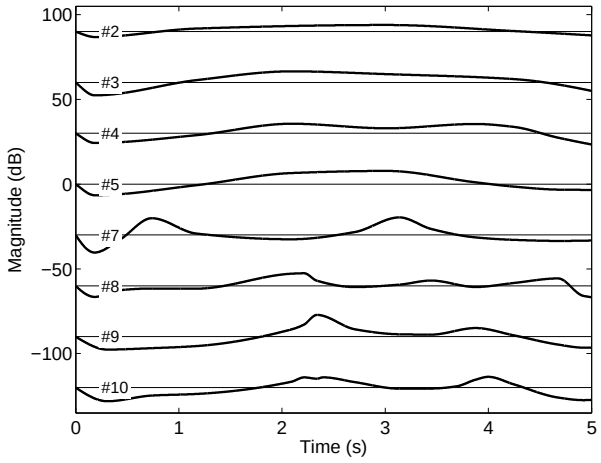


Figure 13: *The modulation envelopes used in Figure 14.*

shape of the peak allows some inaccuracy without affecting the beating frequency. Finally, the proposed method can be used to modify partial envelopes in audio signals, which can be any kind of signals including recorded instrument and synthetic tones.

5. ACKNOWLEDGMENTS

This work was supported by the Nokia Foundation. The author would like to thank Prof. V. Välimäki and Dr. B. Bank for their comments related to this work.

6. REFERENCES

[1] G. Weinreich, "Coupled piano strings," *Journal of the Acoustical Society of America*, vol. 62, no. 6, pp. 1474–1484, 1977.

[2] B. Capleton, "False beats in coupled piano string unisons," *Journal of the Acoustical Society of America*, vol. 115, no. 2, pp. 885–892, 2004.

[3] J. O. Smith III, "Physical modeling using digital waveguides," *Computer Music Journal*, vol. 16, no. 4, pp. 74–91, 1992.

[4] D. A. Jaffe and J. O. Smith III, "Extensions of the Karplus-Strong plucked-string algorithm," *Computer Music Journal*, vol. 7, no. 2, pp. 76–87, 1983.

[5] B. Bank, "Physics-based sound synthesis of the piano," M.S. thesis, Budapest University of Technology and Economics, Budapest, Hungary, 2000.

[6] B. Bank, V. Välimäki, L. Sujbert, and M. Karjalainen, "Efficient physics based sound synthesis of the piano using DSP methods," in *European Signal Processing Conference*, Tampere, Finland, 2000, pp. 2225–2228.

[7] B. Bank, "Accurate and efficient method for modeling beating and two-stage decay in string instrument synthesis," in *MOSART Workshop on Current Research Directions in Computer Music*, Barcelona, Spain, 2001, pp. 124–133.

[8] B. Bank and L. Sujbert, "On the nonlinear commuted synthesis of the piano," in *Proceedings of the 5th International Conference on Digital Audio Effects*, Hamburg, Germany, 2002, pp. 175–180.

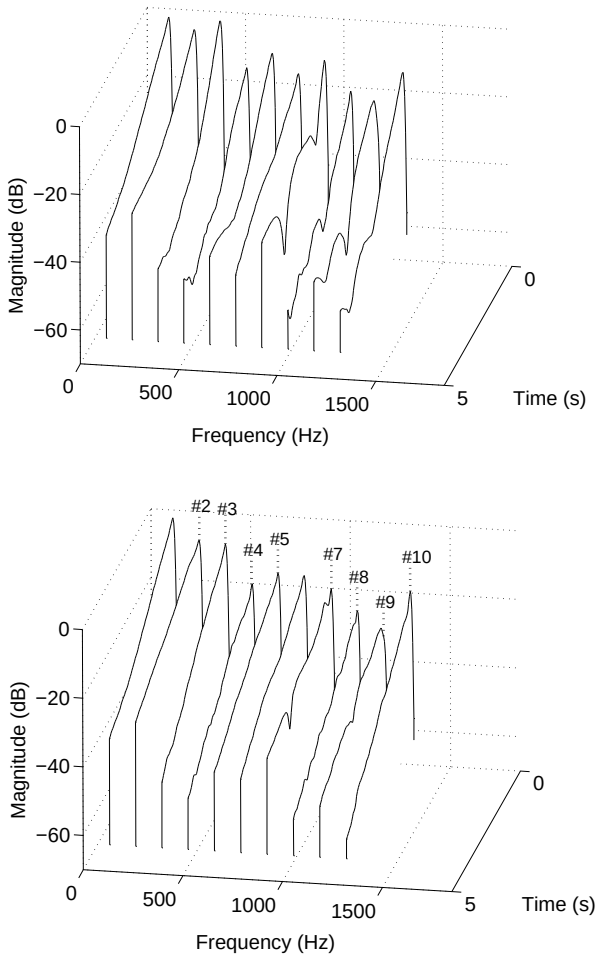


Figure 14: *(Top) The partial envelopes extracted with STFT from the original recorded piano tone (f_0=129.1 Hz, key C_3), and (bottom) the partial envelopes of a modified tone, where partials 2, 3, 4, 5, 7, 8, 9, and 10 have been cancelled with the beating equalizer by using inverse partial envelopes as modulation signals, as seen in Figure 13.*

[9] J. Rauhala, H.-M. Lehtonen, and V. Välimäki, "Toward next-generation digital keyboard instruments," *IEEE Signal Processing Magazine*, vol. 24, no. 2, pp. 12–20, 2007.

[10] H. Järveläinen and M. Karjalainen, "Perception of beating and two-stage decay in dual-polarization string models," in *Proceedings of International Symposium on Musical Acoustics*, Mexico City, Mexico, 2002, pp. 1–10.

[11] P. Regalia and S. Mitra, "Tunable digital frequency response equalization filters," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 35, no. 1, pp. 118–120, 1987.

[12] U. Zölzer, *Digital Audio Signal Processing*, Wiley & Sons, 1997.

[13] J. N. Mourjopoulos, E. D. Kyriakis-Bitzaros, and C. E. Goutis, "Theory and real-time implementation of time-varying digital audio filters," *Journal of the Audio Engineering Society*, vol. 38, no. 7–8, pp. 523–536, 1990.

[14] V. Välimäki and T. I. Laakso, "Suppression of transients in variable recursive digital filters with a novel and efficient cancellation method," *IEEE Transactions on Signal Processing*, vol. 46, no. 12, pp. 3408–3414, 1998.

[15] J. Rauhala and V. Välimäki, "Tunable dispersion filter design method for piano synthesis," *IEEE Signal Processing Letters*, vol. 13, no. 5, pp. 253–256, 2006.

[16] J. Rauhala, H.-M. Lehtonen, and V. Välimäki, "Multi-ripple loss filter for waveguide piano synthesis," in *Proc. International Computer Music Conference*, Barcelona, Spain, 2005, pp. 729–732.

[17] T. I. Laakso, V. Välimäki, M. Karjalainen, and U. K. Laine, "Splitting the unit delay - tools for fractional delay filter design," *IEEE Signal Processing Magazine*, vol. 13, no. 1, pp. 30–60, 1996.

[18] J. Rauhala and V. Välimäki, "Parametric excitation model for waveguide piano synthesis," in *Proc. 2006 IEEE International Conference on Acoustics, Speech, and Signal Processing*, Toulouse, France, 2006, pp. 157–160.

[19] D. Rocchesso and F. Scalcon, "Bandwidth of perceived inharmonicity for physical modeling of dispersive strings," *IEEE Transactions on Speech and Audio Processing*, vol. 7, no. 5, pp. 597–601, 1999.

SIMPLIFIED, PHYSICALLY-INFORMED MODELS OF DISTORTION AND OVERDRIVE GUITAR EFFECTS PEDALS

David T. Yeh, Jonathan S. Abel and Julius O. Smith

Center for Computer Research in Music and Acoustics (CCRMA)
Stanford University, Stanford, CA
[dtyeh|abel|jos]@ccrma.stanford.edu

ABSTRACT

This paper explores a computationally efficient, physically informed approach to design algorithms for emulating guitar distortion circuits. Two iconic effects pedals are studied: the "Distortion" pedal and the "Tube Screamer" or "Overdrive" pedal. The primary distortion mechanism in both pedals is a diode clipper with an embedded low-pass filter, and is shown to follow a nonlinear ordinary differential equation whose solution is computationally expensive for real-time use. In the proposed method, a simplified model, comprising the cascade of a conditioning filter, memoryless nonlinearity and equalization filter, is chosen for its computationally efficient, numerically robust properties. Often, the design of distortion algorithms involves tuning the parameters of this filter-distortion-filter model by ear to match the sound of a prototype circuit. Here, the filter transfer functions and memoryless nonlinearities are derived by analysis of the prototype circuit. Comparisons of the resulting algorithms to actual pedals show good agreement and demonstrate that the efficient algorithms presented reproduce the general character of the modeled pedals.

1. INTRODUCTION

Guitarists tend to feel that digital implementations of distortion effects sound inferior to the original analog gear. This work attempts to provide a more accurate simulation of guitar distortion and a physics based method for designing the algorithm according to the virtual analog approach[1, 2].

Often guitar effects are digitized from a high level understanding of the function of the effect [3, 4]. This work describes the results of a more detailed, physical approach to model guitar distortion. This approach has been adopted previously in the context of generating tube-like guitar distortion [5], not to model a specific effect as done here. This approach starts with the equations that describe the physics of the circuit and is an alternative to obtaining the static transfer curves of a nonlinear system by measurement [6].

Many digital distortion pedals feature pre- and post-distortion filters surrounding a saturating nonlinearity. The filters are commonly multiband (three or four bands) parametric filters that are tuned to taste.

An analysis of the circuits shows that analog solid-state circuits tend to use low-order filters. To keep costs down, circuits are designed with minimal component count, which limits filter order. For the purpose of distortion effect modeling, the frequency range of interest is from just above DC to 20 kHz. Features in the frequency domain above 20 kHz can be ignored, also contributing to low-order filters. Frequency features below 20 Hz must be retained, however, because intermodulation due to mixing of subsonic components with audio frequency components is noticeable in the audio band.

Stages are partitioned at points in the circuit where an active element with low source impedance drives a high impedance load. This approximation is also made with less accuracy where passive components feed into loads with higher impedance. Neglecting the interaction between the stages introduces magnitude error by a scalar factor and neglects higher order terms in the transfer function that are usually small in the audio band.

The nonlinearity may be evaluated as a nonlinear ordinary differential equation (ODE) using numerical techniques [7, 8]. However, the solution of nonlinear ODEs is computationally intensive, and the differences are subtle. Therefore in this work, the nonlinearity is approximated by a static nonlinearity and tabulated. This can be justified on perceptual grounds.

It is well known that nonlinearities cause an expansion of bandwidth that may lead to aliasing if the sampling rate is insufficiently high [3]. Consequently typical digital implementations of distortion upsample by a factor of eight or ten, process the nonlinearities, and downsample back to typical audio rates[3, 9]. Frequency content tends to roll off with increasing frequency, and remaining aliases at oversampling factors of eight or above tend to be masked by the dense spectrum of guitar distortion.

Because the filters in this work are derived from analog prototypes, upsampling also increases the audio band accuracy of the discretization by bilinear transform. An alternate approach would be to design low order filters so that the response at Nyquist matches the continuous time transfer function [10, 11].

The following is an analysis of the stages in two typical distortion pedals.

2. FUNDAMENTAL TOOLS

2.1. SPICE simulation

For circuits that are difficult to analyze, SPICE simulation provides detailed numerical analysis. DC analysis in SPICE performs static sweeps of voltage or current sources to measure memoryless transfer curves. AC analysis finds the frequency response of a circuit linearized about an operating point. These responses can be imported into Matlab and converted to digital filters as in [1]. SPICE also serves as a reference solver for numerical solutions of the time domain response for nonlinear ODEs.

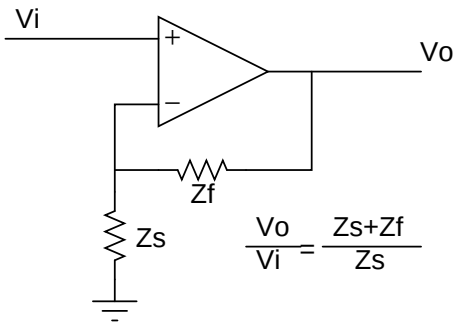


Figure 1: *Non-inverting op amp gain*

2.2. Continuous time pole-zero analysis

Linear circuits are described by rational transfer functions. For most low-cost audio circuits such as guitar effects, the transfer functions are typically low order. The poles and zeros can be identified on a log-frequency plot of magnitude in dB. In dB, it can be seen that the magnitude contributions of poles subtract and the magnitude contributions of zeros add. For the low pass filter, at the pole frequency, the magnitude is 3 dB lower than at its low frequency asymptote. For the high pass filter, the magnitude at the pole frequency is 3 dB lower that at its high frequency asymptote. Therefore, well separated pole and zero frequencies can be identified from the decibel magnitude response by looking for the 3-dB points. These frequencies can then be used to reconstruct the rational expression for the transfer function.

2.3. Analysis of operational amplifier circuits

Transfer functions can be easily found analytically for circuits with operational amplifiers (op amps).

2.3.1. Ideal op amp approximation

The ideal op amp approximation states that if negative feedback is present,

1. $V_+ = V_-$,
2. $I_+ = I_- = 0$

where V_+ is the voltage at the $+$ terminal of the op amp and V_-, the voltage at the $-$ terminal. I_+ and I_- are the currents flowing into the two terminals. These conditions do not hold if negative feedback is not present, for example, if V_o is not connected to V_- or if the op amp output is close to the supply voltages, causing it to clip.

2.3.2. Non-inverting configuration

An example of this analysis is done for the non-inverting op amp configuration shown in Fig. 1. The ideal op amp rule gives $V_- = V_i$, so the current through Z_s is $I_s = V_i/Z_s$. Because $I_- = 0$, all the current flows across Z_f, so $V_0 = V_i + I_s Z_f = V_i + V_i/Z_s$. After algebraic manipulation, the transfer function is found to be $\frac{V_o}{V_i} = \frac{Z_s+Z_f}{Z_s}$. This results in a continuous time transfer function if complex impedances are used for Z_f and Z_s. Writing

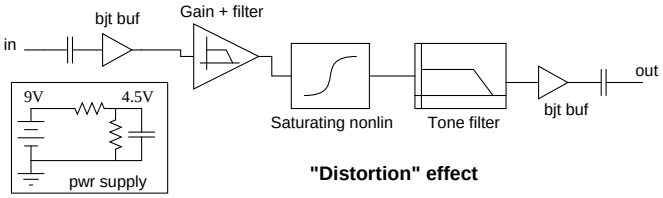


Figure 2: *Block diagram of Distortion pedal.*

it in the form shown in (1) allows the poles and zeros to be seen more easily:

$$A_v(s) = \frac{Z_f}{Z_s}\left(\frac{Z_s}{Z_f} + 1\right) \tag{1}$$

2.4. Bilinear Transform of low order transfer functions

Once a continuous time transfer function is obtained either by analysis or by inspection of the magnitude response, the bilinear transform can be used to digitize this filter. First- and second-order continuous time systems are common, so their mappings are given below.

The continuous time system,

$$H(s) = \frac{b_n s^n + ... + b_1 s + b_0}{a_n s^n + ... + a_1 s + a_0}, \tag{2}$$

results in

$$H(z) = \frac{B_0 + B_1 z^{-1} + ... + B_n z^{-n}}{A_0 + A_1 z^{-1} + ... + A_n z^{-n}}, \tag{3}$$

where for a second order system, coefficients of $H(z)$ are

$$B_0 = b_0 + b_1 c + b_2 c^2, \tag{4}$$
$$B_1 = 2b_0 - 2b_2 c^2, \tag{5}$$
$$B_2 = b_0 - b_1 c + b_2 c^2, \tag{6}$$
$$A_0 = a_0 + a_1 c + a_2 c^2, \tag{7}$$
$$A_1 = 2a_0 - 2a_2 c^2, \tag{8}$$
$$A_2 = a_0 - a_1 c + a_2 c^2, \tag{9}$$

and for a first-order system, coefficients of $H(z)$ are

$$B_0 = b_0 + b_1 c, \tag{10}$$
$$B_1 = b_0 - b_1 c, \tag{11}$$
$$A_0 = a_0 + a_1 c, \tag{12}$$
$$A_1 = a_0 - a_1 c. \tag{13}$$
$$\tag{14}$$

Here $c = 2/T$ is chosen as typical for the bilinear transform.

3. CIRCUIT ANALYSIS OF DISTORTION PEDAL

The block diagram of the Boss DS-1 Distortion pedal [12] is shown in Fig. 2. It is basically gain with a saturating nonlinearity sandwiched between filters. However, distortion from the bipolar transistor (BJT) emitter follower buffers and first gain stage are not negligible.

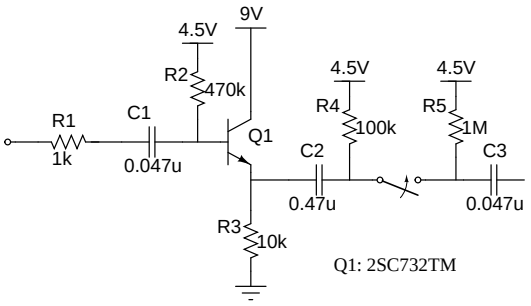


Figure 3: *Input buffer: Emitter follower circuit.*

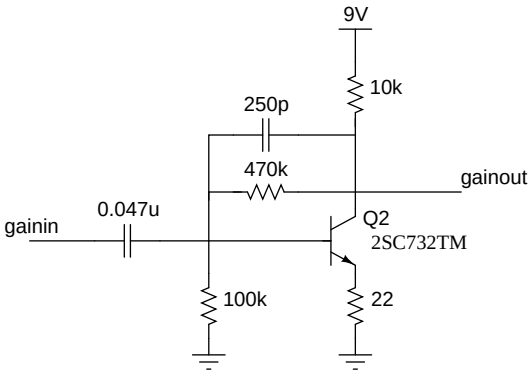


Figure 4: *BJT transimpedance gain*

3.1. Emitter Follower buffers

A typical guitar pedal has an emitter follower (Fig. 3) at the input to buffer the signal from the guitar pickups, and another similar emitter follower at the output to drive the cable and subsequent load. The emitter follower topology is nominally linear in operation and flat in frequency response in the audio band. Typically it is used in conjunction with high pass filters, whose cutoff frequency can be determined from the resistance and capacitance values. Here it is 3 Hz. The stage can be implemented as cascaded low order high pass filters . Implementation of high pass filters is straightforward with the bilinear transform method of digitizing an analog prototype as described in Section 2.

3.2. Single bipolar transistor transimpedance gain stage

Gain can be provided by a single bipolar junction transistor (BJT) in a transimpedance gain topology shown in Fig. 4.

The frequency response is found from SPICE and digitized by finding the continuous time poles and zeros, forming the transfer function and taking the bilinear transform. This stage shows 36 dB of bandpass gain (Fig. 5). There are two zeros at DC, one pole at 3 Hz, one pole at 600 Hz, and another at 72 kHz, which is ignored because it is well outside the audio band. A transfer function is formed directly in (15):

$$H(s) = \frac{s^2}{(s+\omega_1)(s+\omega_2)}, \tag{15}$$

where the numerator is the product of two zeros, s, and the denominator is the product of the poles at $\omega_1 = 2\pi 3$ and $\omega_1 = 2\pi 600$.

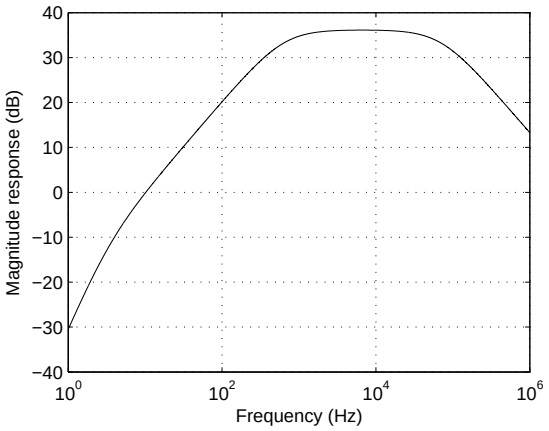


Figure 5: *Frequency response of BJT stage*

The bilinear transform applied to $H(s)$ with a sampling rate $f_s = 44.1$ kHz gives a second order digital filter whose coefficients can be found using (9).

This stage introduces significant nonlinearity at large inputs, but this is neglected for now.

3.3. Op amp gain stage

Non-inverting op amp "buffers" are common in guitar circuits because they minimize loading on the preceding stage. To analyze the circuit in Fig. 6 impedances are used in (1). The final transfer function in factored form is given by (16).

$$H(s) = \frac{(s+\frac{1}{R_t C_c})(s+\frac{1}{R_b C_z}) + \frac{s}{R_b C_c}}{(s+\frac{1}{R_t C_c})(s+\frac{1}{R_b C_z})} \tag{16}$$

where $R_t = D{\cdot}100\text{k}\Omega$, $R_b = (1-D)100\text{k}\Omega{+}4.7\text{k}\Omega$, $C_z = 1\mu$F, and $C_c = 250$pF. Capacitor C_z blocks DC to prevent the output from saturating in the presence of DC offset, while C_c stabilizes the op amp and contributes a low pass pole. D ranges between $(0, 1)$ and is the value of the "DIST" knob that controls the amount of gain before saturation and therefore the intensity of the distortion.

The frequency response is shown in Fig. 7 for values of D from 0 to 1 in increments of 0.1. This is a second-order stage than can be digitized directly by the bilinear transform, forming a second-order section with variable coefficients. The frequency response of this stage depends on the "DIST" knob. Notice that the frequency response at half the audio sampling rate, $|H(f = 22050)|$, is not zero and considerable warping will take place without oversampling or the filter design method by Orfanidis [10].

This transfer function can be discretized by the bilinear transform, (9), which also preserves the mapping of the "DIST" parameter.

The op amp provides the main nonlinearity of the Distortion effect. To first order, the op amp hard clips the signal at $V_{dd}/2$. In reality the op amp response is much slower because it is open loop and needs to recover from overdrive. It is also typically asymmetrical in behavior, leading to significant even-order harmonics where otherwise only odd-order harmonics are expected. Refinements of the op amp clipping model can be based upon the macromodeling technique as done in SPICE to speed up simulations [13]. A

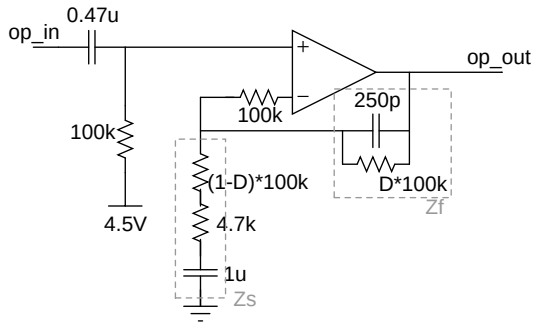


Figure 6: *Operational amplifier gain stage*

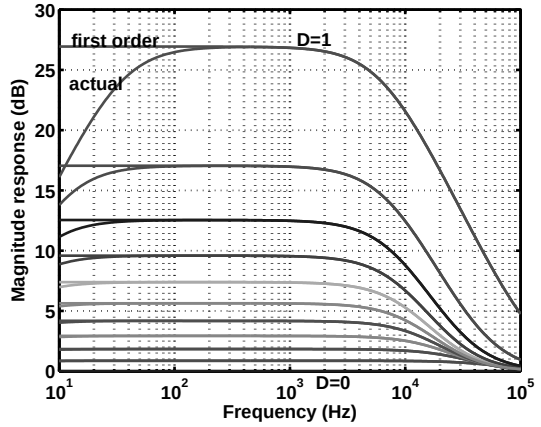


Figure 7: *Frequency response of op amp gain stage,* $D = 0 : 0.1 : 1$

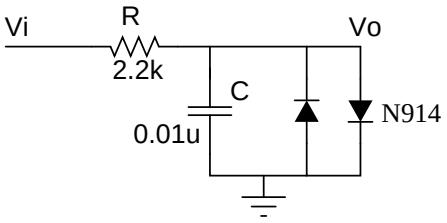


Figure 8: *RC low pass filter with diode limiter*

black box approach, the macromodeling technique emulates the input/output behavior of the op amp instead of simulating the behavior of its internal devices.

3.4. Diode clipper

Following the op amp clipper is a RC low pass filter with a diode limiter across the capacitor (Fig. 8). The diode clipper limits the voltage excursion across the capacitor to about a diode drop in either direction about signal ground.

The model of the pn diode is

$$I_d = I_s(e^{V/V_t} - 1), \tag{17}$$

where the reverse saturation current I_s, and thermal voltage V_t of

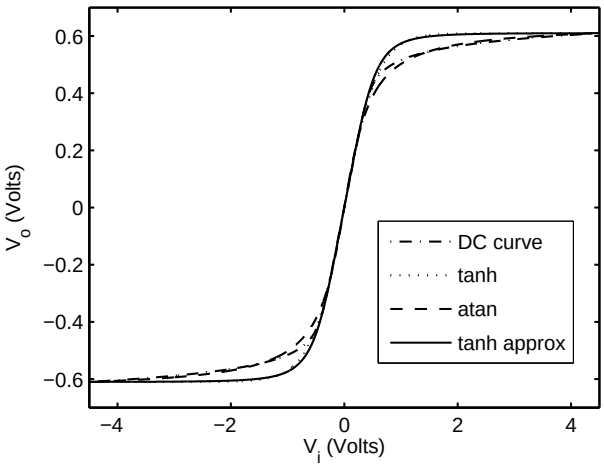


Figure 9: *Static nonlinear functions compared: tabulated, tanh, arctan, approximation to tanh*

the device are model parameters that can be extracted from measurement.

The nonlinear ODE of the diode can be derived from Kirchhoff's laws:

$$\frac{\mathrm{d}V_o}{\mathrm{d}t} = \frac{V_i - V_o}{RC} - 2\frac{I_s}{C}(\sinh(V_o/V_t)), \tag{18}$$

where V_i, V_o are the input and output signals respectively.

This is not a memoryless nonlinearity because it is a low-pass filter whose pole location changes with voltage. Fig. 10 depicts the input-output characteristic, which can be described as a "clipping" function, along with various analytic approximations based on hyperbolic tangent and arctangent. At high amplitude levels, the differences between different clipping functions is subtle.

For efficiency, this nonlinearity is approximated as static, and the DC transfer curve is computed by setting $\frac{\mathrm{d}V_o}{\mathrm{d}t} = 0$ in (18), and tabulating the function $V_o = f(V_i)$ by Newton iteration. A nonuniform sampling of the input to output transfer curve is used that utilizes a constant error percentage or signal to quantization noise ratio. The rationale for this is that at small amplitudes, the curve is most linear with the highest gain, and most susceptible to quantization noise. At high levels, the nonlinearity is compressive, reducing the gain and quantization error. A logarithmic sampling with a floor about zero is chosen. Linear interpolation is used to further reduce quantization noise.

Alternatively an approximation such as

$$\frac{x}{(1 + |x|^n)^{1/n}} \tag{19}$$

can be used to compute the nonlinearity. This formula (19) well approximates hyperbolic tangent when $n = 2.5$. The transfer curve of the tabulated function is compared with that of tanh, arctan, and (19) in Fig. 9. The curves are normalized so that the slope about $V_i = 0$ matches visually and V_o at the extremes match. The formula (19) can be seen to be a good approximation of tanh. Arctan looks like a close approximation to the actual DC nonlinearity but it is not as linear about $V_i = 0$. The approximation (19) has the advantage of an additional parameter n that can be varied to better

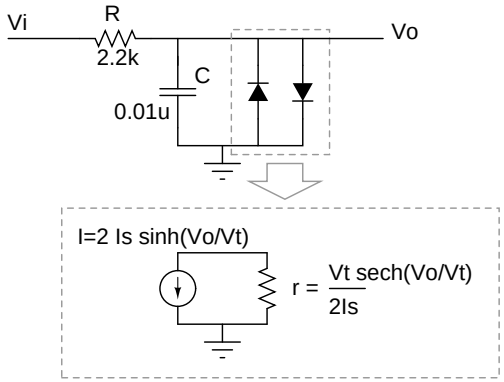


Figure 10: *Small signal approximation of diode clipper*

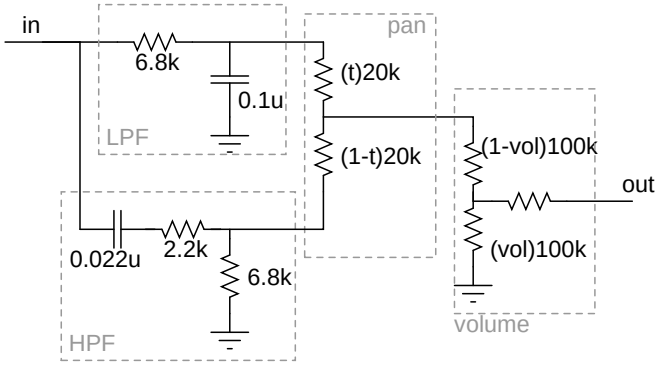


Figure 11: *Tone circuit of Distortion pedal*

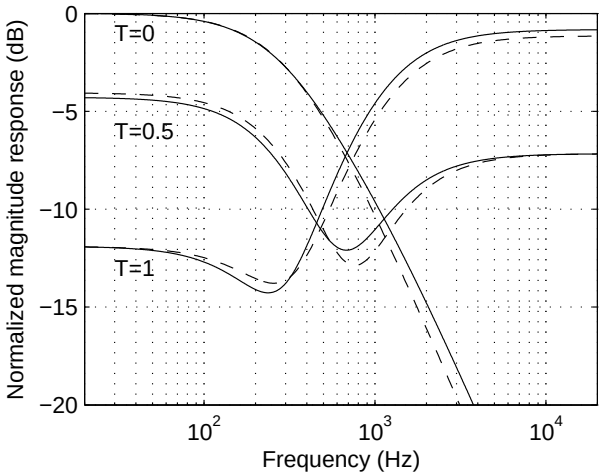


Figure 12: *Distortion pedal tone circuit frequency response. Solid line is actual. Dashed line is digitized implementation.*

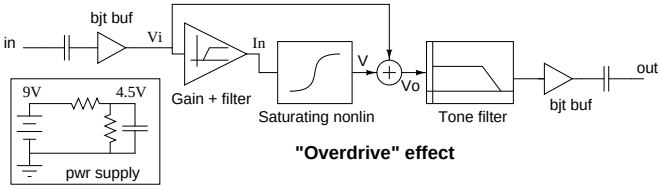


Figure 13: *Block diagram of Overdrive pedal.*

match the actual function. In this work, the tabulated nonlinearity is chosen.

3.5. Tone stage

The tone stage (Fig. 11) is a highly interconnected passive network that cannot be accurately separated. However, an analysis of the circuit shows its design intent, and the error due to separating the blocks is less than that due to component tolerance in an actual circuit.

This circuit involves a fade between high pass filter and low pass filter blocks. The fading affects the cutoff frequencies of the filters, but this effect is small. A digitization of this circuit can capture the essence of its operation, which is a loudness boost: a V-shaped equalization as commonly observed for tone circuits intended for electric guitars[5, 1].

A full analysis is straightforward but tedious, so AC analysis is performed in SPICE, and the corner frequencies found graphically. The weightings for the fade are also determined by simulation. The high pass corner frequency is $f_{hpf} = 1.16$ kHz and the low pass corner frequency is $f_{lpf} = 320$ Hz.

This is implemented digitally as a weighted sum of first-order high pass and low pass filters discretized by the bilinear transform rather than discretizing the complete transfer function. This simplification eliminates time-varying filters and the computation to update the coefficients, using static coefficients instead. Modeling a user controlled parameter with greater accuracy is unnecessary because a user would not likely notice the difference in filter response.

The magnitude response of the original circuit is compared with the Matlab approximation in Fig. 12. The responses are very similar with < 1 dB error in most cases.

4. CIRCUIT ANALYSIS OF OVERDRIVE PEDAL

The block diagram of an overdrive pedal, specifically the Ibanez Tube Screamer, is given in Fig. 13[14]. It is characterized by high pass filters, followed by the summation of a high-pass filtered and clipped signal summed with the input signal. This is followed by low-pass tone filtering and a high pass in the output buffer. The following is an analysis of the circuit in rigor

4.1. High pass filters

The first stages of the overdrive pedal are high pass filters with the following cutoff frequencies: $f_{c1} = 15.9$ Hz, $f_{c2} = 15.6$ Hz.

4.2. Non-inverting op amp with diode limiter

The non-inverting op amp (Fig. 14) of the overdrive pedal is similar to that of the distortion except the diode limiter is now across Z_f. The diode limiter essentially limits voltage excursion across the op amp keeping it within ideal op amp conditions. The voltage at the minus input of the op amp is then the same as that on the

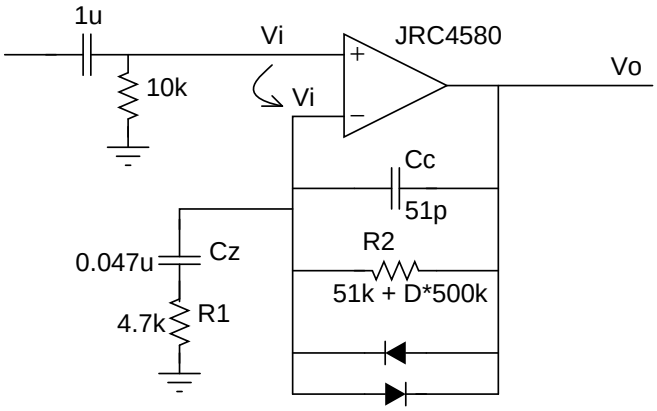


Figure 14: *Clipping stage of overdrive pedal.*

plus terminal. This generates a current across Z_s,

$$I_n = \frac{V_{neg}}{Z_s} = V_i \frac{s}{R_1(s+\omega_z)}, \tag{20}$$

where $\omega_z = (R_1 C_z)^{-1}$, $R_1 = 4.7\text{k}\Omega$, $C_z = 0.047\mu$F. I_n flows through the components connected between the minus terminal and the output of the op amp. Circuit analysis produces the following equation:

$$I_n = \frac{V_o - V_i}{R_2} + C_c \frac{\mathrm{d}}{\mathrm{d}t}(V_o - V_i) + 2I_s \sinh(\frac{V_o - V_i}{V_t}) \tag{21}$$

Making a variable substitution $V = V_o - V_i$ yields,

$$\frac{\mathrm{d}V}{\mathrm{d}t} = \frac{I_n}{C_c} - \frac{V}{R_2 C_c} - \frac{2I_s}{C_c} \sinh(V/V_t), \tag{22}$$

where $C_C = 51$pF, $R_2 = 51\text{k} + D500\text{k}$, and $D \in (0, 1)$, controlling the intensity of the overdrive. It can be seen that this ODE is the same as that for the Distortion pedal, (18), when I_n is replaced by V_i/R.

The arithmetic introduced by the variable substitution can be described in block diagram form as depicted in Fig. 13. The essence of the overdrive circuit is the summation of the input signal with the input filtered and clipped. The above variable substitution is solved for V_o:

$$V_o = V + V_i, \tag{23}$$

where V is obtained by solving (22).

4.3. Tone stage

The tone stage (Fig. 15) can also be analyzed according to ideal op amp rules. The algebra is complicated, but the result is

$$\frac{V_o}{V_i} = \frac{(R_l R_f + Y)}{Y R_s C_s} \frac{s + W\omega_z}{(s+\omega_p)(s+\omega_z) + Xs}, \tag{24}$$

where

$$W = \frac{Y}{R_l R_f + Y},$$

$$X = \frac{R_r}{R_l + R_r} \frac{1}{(R_z + R_l \| R_r) C_z},$$

$$Y = (R_l + R_r)(R_z + R_l \| R_r),$$

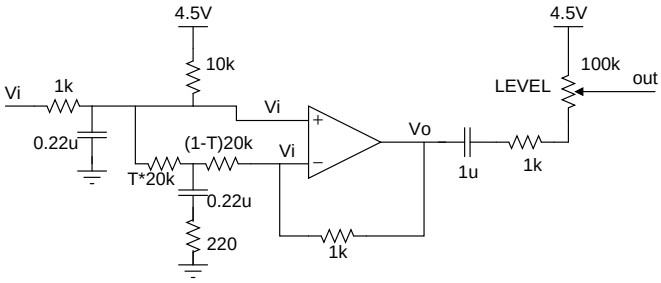


Figure 15: *Overdrive tone circuit.*

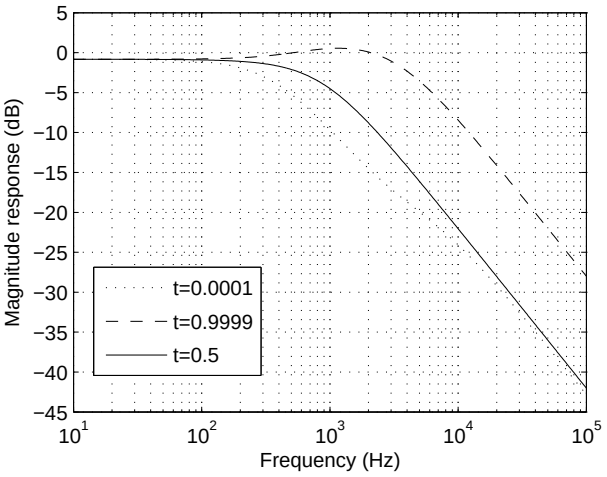


Figure 16: *Overdrive tone circuit frequency response for T = 0, 0.5, 1.*

$\omega_z = 1/(C_z(R_z + R_l \| R_r))$, $\omega_p = 1/(C_s(R_s \| R_i))$, $R_f = 1\text{k}$, $R_r = (1-T)20\text{k}$, $R_l = T20\text{k}$, $R_z = 220$, $C_z = 0.22\mu$F, $R_i = 10\text{k}$, $R_s = 1\text{k}$, $C_s = 0.22\mu$F, and $T \in (0, 1)$ controls the cutoff frequency of the low pass.

This is a second-order transfer function with variable coefficients. Fig. 16 shows the essentially low-pass character of the magnitude response.

5. RESULTS

Actual Distortion and Overdrive pedals are compared to the digital emulations for a 220 Hz sine signal with amplitude of 200 mV, and an exponential sine sweep. The settings on the actual pedal are adjusted until the spectrum resembles that of the digital version for the sine input. Adjustments were made approximately to match the difference in magnitude of the first two harmonics, and to match the position of notches in the frequency domain.

The time waveforms and magnitude spectra for the single-frequency excitation are shown in Figs. 17–20. The sinusoidal sweeps are represented by a log-frequency spectrogram [15] in Figs. 21–24 with 80-dB dynamic range.

The waveforms show a general similarity. The spectrograms indicate that frequency equalization is close. The measured spectra exhibit a strong even-order nonlinearity that is not modeled in the digital implementation. The emulated versions using the simplified algorithms in both cases sound slightly brighter than the

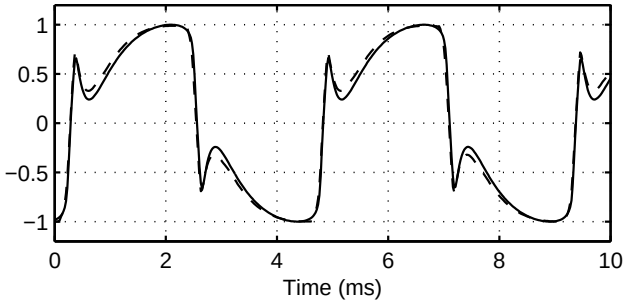


Figure 17: *Time response to 220 Hz sine, measured distortion pedal (dashed) and algorithm (solid)*

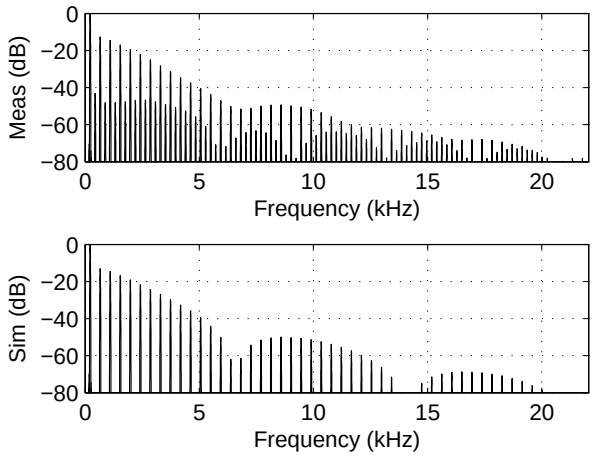


Figure 18: *Normalized spectrum of response to 220 Hz sine, distortion pedal (top), algorithm (bottom)*

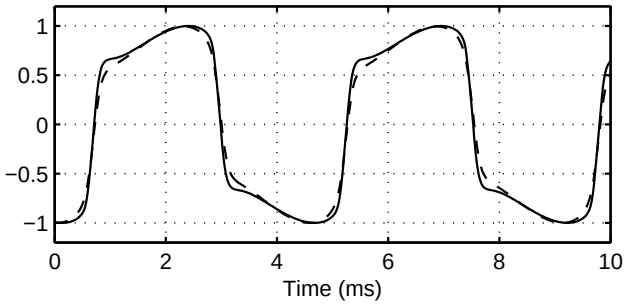


Figure 19: *Time response to 220 Hz sine, measured overdrive pedal (dashed) and algorithm (solid)*

actual pedals, possibly due to the lack of even-order nonlinearity and a difference in equalization..

The digitally emulated result also deviates from the measured one because there was no attempt to calibrate the model to the actual pedal with its particular component values and parameter settings. It is more representative of a circuit whose components are exactly the values as in the schematic.

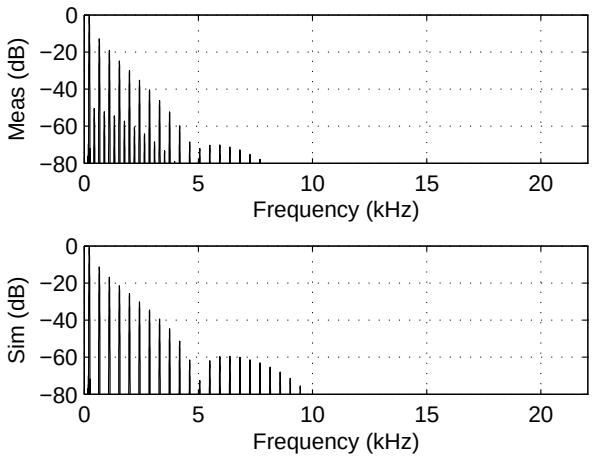


Figure 20: *Normalized spectrum of response to 220 Hz sine, overdrive pedal (top), algorithm (bottom)*

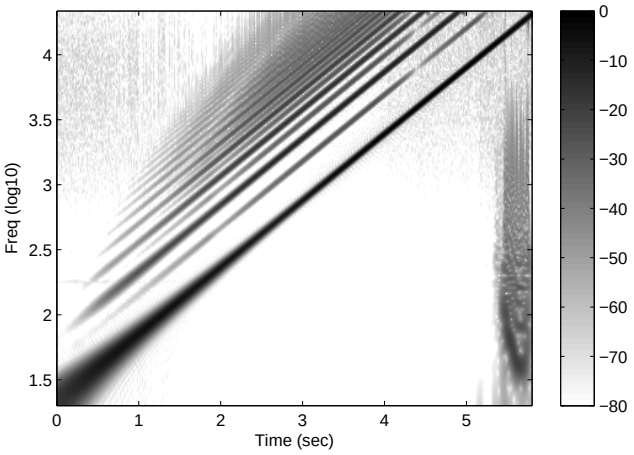


Figure 21: *Measured distortion pedal, sine sweep log spectrogram*

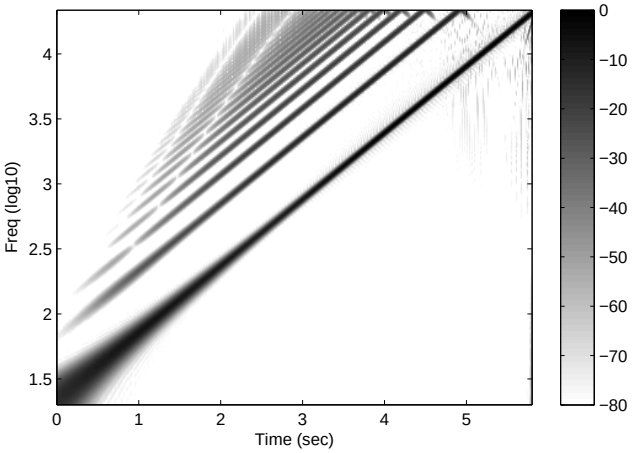


Figure 22: *Distortion algorithm, sine sweep log spectrogram*

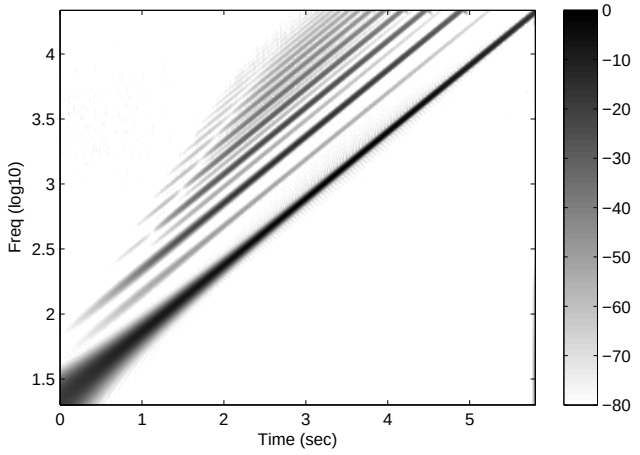


Figure 23: *Measured overdrive pedal, sine sweep log spectrogram*

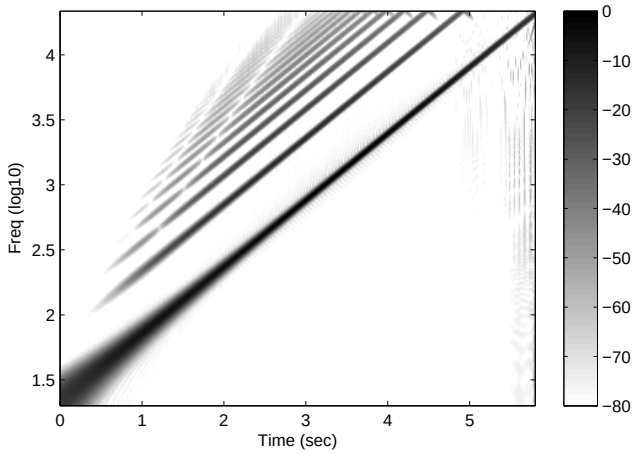


Figure 24: *Overdrive algorithm, sine sweep log spectrogram*

6. CONCLUSIONS

The simplified, physically informed approach enables the design of distortion algorithms that emulate the behavior of analog prototypes. A first-pass design with no tuning is able to reproduce the salient characteristics of the effect.

While the result is not an exact emulation of the analog implementation, it provides a procedural basis for the design of a distortion algorithm, and a starting point from which further tuning can be done. The computational power needed is comparable to that available in commercially available guitar effects boxes because of the similar architecture comprising oversampling, low order filters, and a tabulated nonlinearity.

In this work, BJT gain stage and op amp clipping behaviors are oversimplified. Nonlinearities are assumed to come from a single symmetrical diode clipper, which is not true under large-signal conditions. Improved models of remaining nonlinearities are the subject of ongoing research.

7. ACKNOWLEDGMENTS

This work has been supported by the NSF and NDSEG graduate fellowships. Thanks to Digidesign for their critique and insightful comments. Thanks to Boss and Ibanez for producing these unique music processing circuits and their impact on the guitar playing world.

8. REFERENCES

[1] D. T. Yeh and J. O. Smith, "Discretization of the '59 Fender Bassman tone stack," in *Proc. of the Int. Conf. on Digital Audio Effects (DAFx-06)*, Montreal, Quebec, Canada, Sept. 18–20, 2006, pp. 1–6.

[2] A. Huovilainen, "Nonlinear Digital Implementation of the Moog Ladder Filter," in *Proc. of the Int. Conf. on Digital Audio Effects (DAFx-04)*, Naples, Italy, Oct. 5–8, 2004.

[3] U. Zölzer, Ed., *DAFX - Digital Audio Effects*, J. Wiley & Sons, 2002.

[4] J. Schimmel, "Using nonlinear amplifier simulation in dynamic range controllers," in *Proc. of the Int. Conf. on Digital Audio Effects (DAFx-03)*, London, UK, Sep. 8-11 2003.

[5] M. Karjalainen, T. Mäki-Patola, A. Kanerva, and A. Huovilainen, "Virtual air guitar," *Journal of the AES*, vol. 54, no. 10, pp. 964–980, Oct. 2006.

[6] S. Möller, M. Gromowski, and U. Zölzer, "A measurement technique for highly nonlinear transfer functions," in *Proc. of the Int. Conf. on Digital Audio Effects (DAFx-02)*, Hamburg, Germany, Sep. 26-28 2002, pp. 203–206.

[7] M. Karjalainen and J. Pakarinen, "Wave digital simulation of a vacuum-tube amplifier," in *IEEE ICASSP 2006 Proc.*, Toulouse, France, 2006, vol. 5, pp. 153–156.

[8] D. T. Yeh, J. Abel, and J. O. Smith, "Simulation of the diode limiter in guitar distortion circuits by numerical solution of ordinary differential equations," in *Proc. of the Int. Conf. on Digital Audio Effects (DAFx-07)*, Bordeaux, France, Sept. 10–15, 2007.

[9] M. Doidic and et al., "Tube modeling programmable digital guitar amplification system," U.S. Patent 5789689, Aug. 16 1998.

[10] S. J. Orfanidis, "Digital parametric equalizer design with prescribed nyquist-frequency gain," *Journal of the AES*, vol. 45, no. 6, pp. 444–455, June 1997.

[11] D. P. Berners and J. S. Abel, "Discrete-time shelf filter design for analog modeling," in *Proc. 115th AES Convention*, New York, Oct. 2003.

[12] Roland Corp., *Boss DS-1 Service Notes*, Dec. 26 1980.

[13] G. R. Boyle, D. O. Pederson, B. M. Cohn, and J. E. Solomon, "Macromodeling of integrated circuit operational amplifiers," *IEEE J. Solid-State Circuits*, vol. 9, pp. 353–364, Dec 1974.

[14] R. G. Keen, "The Technology of the Tube Screamer," Available at http://www.geofex.com/fxtech.htm, Accessed Mar 22, 2007.

[15] J. C. Brown, "Calculation of a constant Q spectral transform," *J. Acoust. Soc. Am*, vol. 89, pp. 425–434, 1991, Matlab code available at http://web.media.mit.edu/ brown/cqtrans.htm.

SIMULATION OF THE DIODE LIMITER IN GUITAR DISTORTION CIRCUITS BY NUMERICAL SOLUTION OF ORDINARY DIFFERENTIAL EQUATIONS

David T. Yeh, Jonathan Abel, and Julius O. Smith

Center for Computer Research in Music and Acoustics (CCRMA)
Stanford University, Stanford, CA
[dtyeh|abel|jos]@ccrma.stanford.edu

ABSTRACT

The diode clipper circuit with an embedded low-pass filter lies at the heart of both diode clipping "Distortion" and "Overdrive" or "Tube Screamer" effects pedals. An accurate simulation of this circuit requires the solution of a nonlinear ordinary differential equation (ODE). Numerical methods with stiff stability – Backward Euler, Trapezoidal Rule, and second-order Backward Difference Formula – allow the use of relatively low sampling rates at the cost of accuracy and aliasing. However, these methods require iteration at each time step to solve a nonlinear equation, and the tradeoff for this complexity must be evaluated against simple explicit methods such as Forward Euler and fourth order Runge-Kutta, which require very high sampling rates for stability. This paper surveys and compares the basic ODE solvers as they apply to simulating circuits for audio processing. These methods are compared to a static nonlinearity with a pre-filter. It is found that implicit or semi-implicit solvers are preferred and that the filter/static nonlinearity approximation is often perceptually adequate.

1. INTRODUCTION

Guitarists tend to feel that digital implementations of distortion effects sound inferior to the original analog gear. This work attempts to provide a more accurate simulation of guitar distortion and a physics-based method for designing the algorithm in the same manner physical modeling is done for acoustic instruments. Guitar effects consists of circuits that are accurately described in the audio frequency band by nonlinear ordinary differential equations (ODEs). Often the circuits are comprised of linear stages that can be efficiently implemented by infinite impulse response (IIR) digital filters. The remaining nonlinear ODEs may need to be solved by a numerical method or other approximation.

The diode clipper circuit with an embedded low-pass filter forms the basis of both diode clipping "Distortion" and "Overdrive" or "Tube Screamer" effects pedals[1]. An accurate simulation of this circuit requires the solution of a nonlinear ordinary differential equation (ODE). Numerical methods with stiff stability, Backward Euler, Trapezoidal Rule, and second order Backward Difference Formula (BDF, also called Gear) allow the use of low sampling rates at the cost of accuracy and aliasing[2]. However, these methods require iteration at each time step to solve a nonlinear equation, and the tradeoff for this complexity must be evaluated against simple explicit methods such as Forward Euler and fourth-order Runge-Kutta, which require very high sampling rates for stability.

1.1. Related work

A nonlinear system can be represented analytically as a Volterra series. There has been work to form finite-order Volterra series for simulating electronics [3, 4]. However, these are interesting only for low order, whereas for highly nonlinear systems, direct simulation by numerical methods is more computationally efficient [5].

Numerical solution of ODEs is a very mature topic in applied mathematics and many sophisticated algorithms exist for improving accuracy and speed [2, 6, 7, 8]. Backward Euler, Trapezoidal Rule, and BDF (called Gear in SPICE) are options to solve the nonlinear ODEs in circuits [9, 10]. Most algorithms are designed with a variable step size (sampling rate) to maintain a specified accuracy. The error is typically specified in the time-domain and is related to an order of the step size. Matlab features a rich suite of ODE solvers [11] that can be used for experimentation and gaining experience with the solution of ODEs.

Although not presented as such, an example of numerical simulation of ODEs for musical application is [12]. The WDF is an alternate implementation of the trapezoidal rule. It is equivalent to trapezoidal rule integration and results in the same iterations being solved because the nonlinearity is expressed in K-variables.[1]

1.2. Error criterion

Most applications for ODE solvers have a different set of requirements than those for audio. The error criterion for general solvers adaptively adjusts the step size to an excessively small value.

A fixed sampling rate is better suited for implementation in real-time audio processing. In addition, borrowing from the field of perceptual audio coding [13], the error specification for audio is best defined perceptually in the short-time frequency domain.

For audio, using a larger step size to reduce computational costs may increase aliasing, but this is tolerable if below the masking threshold of the desired audio signal. Also, the original analog electronics have a relatively high noise floor which would mask low level aliasing.

In this paper, the audio band is defined to be 0–20 kHz, where a match to the accurate solution of the ODE is desired. Subsonic frequencies are included because they may mix through the nonlinearity and cause perceptible amplitude modulation of the output. High frequencies are assumed to be sufficiently low due to roll-off of typical spectra that mixing products are negligible.

[1] The term "K-variable" means "Kirchoff variable," such as a voltage or current. In contrast, WDFs use "W-variables" ("wave variables"). K-variables can be converted to W-variables and vice versa by means of a two-by-two matrix multiply as in (8).

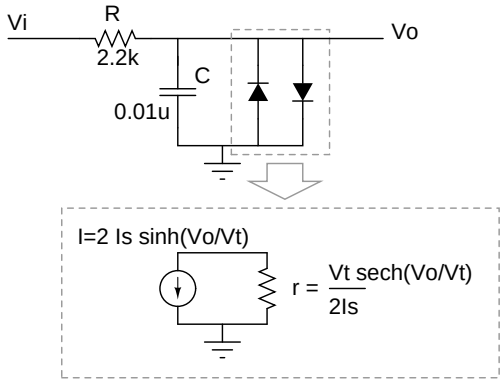


Figure 1: *Small signal approximation of diode clipper*

1.3. Diode clipper equation

Often nonlinearities for virtual analog modeling are assumed to be memoryless. The derivation below proves this to be an incorrect assumption, although cascading filters with a memoryless nonlinearity may be used as a perceptually close approximation. That the nonlinearity has memory is also suggested in [14], where the measurement technique to find the nonlinear transfer curves in a tube amp does not find a smooth nonlinearity, but rather a noisy one due to hysteresis.

The diode clipper in guitar circuits is typically a RC low-pass filter with a diode limiter across the capacitor (Fig. 1). The diode clipper limits the voltage excursion across the capacitor to about a diode drop in either direction about signal ground.

The model of the pn diode is

$$I_d = I_s(e^{V/V_t} - 1), \tag{1}$$

where the reverse saturation current I_s, and thermal voltage V_t of the device are model parameters that can be extracted from measurement.

The nonlinear ODE of the diode can be derived from Kirchhoff's laws:

$$\frac{\mathrm{d}V_o}{\mathrm{d}t} = \frac{V_i - V_o}{RC} - 2\frac{I_s}{C}\sinh(V_o/V_t), \tag{2}$$

where V_i, V_o are the input and output signals respectively.

This small-signal interpretation (Fig. 1) of this circuit contradicts the assumption of a memoryless nonlinearity because it yields a low-pass filter whose pole location changes with voltage.

2. NUMERICAL METHODS

2.1. Basic methods

The basic methods solve equations of the form

$$\frac{\mathrm{d}v}{\mathrm{d}t} = \dot{v} = f(t, v) \tag{3}$$

where the system state v is, in general, a vector, and $f(t, v)$ is a nonlinear function which encompasses any input $u(t)$ to the system. Time, t, is the independent variable of integration for ODEs that describe circuits.

In the case of a linear constant-coefficient differential equation, (3) can be written in state space form:

$$\dot{v}(t) = Av(t) + Bu(t) \tag{4}$$

where the eigenvalues of A are the poles of the system. Linear filters are efficient solvers of this special case of ODEs.

Explicit methods are those whose output depend only on state from previous time steps. Implicit formulas may depend on current state and require iteration if the ODE is nonlinear. Newton's method is the most popular solver, in part because it is scalable to higher dimensions. For single dimensional equations, bisection or bracketing provides predictable convergence [2].

In the literature for numerical methods, the methods are notated with subscripts denoting the time index and h for step size (sampling period). Here the methods are presented using square brackets to denote the time index and T for step size as for digital filters.

2.1.1. Forward Euler (FE)

$$v[n] = v[n-1] + T\dot{v}[n-1], \tag{5}$$

where $v[n]$ is the system state at discrete time n, and T is the sampling interval. This is an explicit, first-order method.

2.1.2. Backward Euler (BE)

$$v[n] = v[n-1] + T\dot{v}[n], \tag{6}$$

Note that this is similar to (5) except this is an implicit equation.

2.1.3. Trapezoidal Rule (TR)

$$v[n] = v[n-1] + \frac{T}{2}\left(\dot{v}[n] + \dot{v}[n-1]\right), \tag{7}$$

Trapezoidal rule is similar to the Euler methods, using instead the average of the derivatives at time n and $n-1$, resulting in a second-order method.

It is known that the trapezoidal rule has the smallest truncation error of any method of order two [10]. It is also equivalent to the discretization of a continuous-time transfer function by the bilinear transform. The trapezoidal rule is also the only practical order-preserving method that does not introduce artificial damping when discretizing continuous time systems.

2.1.4. Wave Digital Filter implementation of Trapezoidal Rule integration

The wave digital filter (WDF) is an alternate implementation of the trapezoidal rule integration where K-variables V, I (values corresponding to physical voltages and currents) are replaced by W-variables a, b by the mapping [12, 15]

$$\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 1 & R_0 \\ 1 & -R_0 \end{bmatrix} \begin{bmatrix} V \\ I \end{bmatrix} \tag{8}$$

Summarizing the approach in [15, 16], to make a nonlinear WDF, one may use (1), which is in the form $I = I(V)$ and substitute into (8) and obtain the wave variables:

$$a = f(V) = V + RI(V) \tag{9}$$

$$b = g(V) = V - RI(V) \tag{10}$$

Given an incident wave a and an invertible $f(V)$, one may find V and then use (10) to find b. Because $I(V)$ is the nonlinear function (1), this requires an iterative method. Note that the iteration involves the K-variable V. Therefore, the resulting iterations are identical to the direct application of the trapezoidal rule to the ODE.

The trapezoidal rule is implied in the use of the bilinear transform to locally discretize capacitors and inductors. The nonlinear WDF can thus be viewed as an alternative implementation of the trapezoidal rule solver for nonlinear ODEs.

The WDF poses the advantage that the nonlinear equations are solved locally for each nonlinear element, minimizing the size of the matrix equation to be inverted, even if the circuit contains many nodes or elements.

2.1.5. Backward Difference Formula

The Backward Difference Formula of order two (BDF2) is commonly used in circuit simulation and deserves mention here. It is a multi-step method that only requires a single function evaluation of (3).

$$v[n] = \frac{4}{3}v[n-1] - \frac{1}{3}v[n-2] + \frac{2T}{3}\dot{v}[n] \tag{11}$$

2.1.6. Runge-Kutta

A popular higher-order one-step method is the explicit fourth-order Runge-Kutta formula (RK4).

$$\begin{aligned}
k_1 &= Tf(n-1, v[n-1]),\\
k_2 &= Tf(n-1/2, v[n-1] + k_1/2),\\
k_3 &= Tf(n-1/2, v[n-1] + k_2/2),\\
k_4 &= Tf(n, v[n-1] + k_3),\\
f(n,v) &= \dot{v}(n,v) = \frac{V_i[n] - v}{RC} - 2\frac{I_s}{C}\sinh(v/V_t),\\
v[n] &= v[n-1] + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6}
\end{aligned}$$

The function evaluations at times in between samples could be inconvenient for digital audio. The only time dependence in the diode clipper ODE is the input voltage. Therefore, RK4 requires input at twice the sampling rate of the output. On the contrary, a method should have a higher output rate than input rate because the nonlinearity causes the output to have a wider bandwidth than the input.

2.1.7. Semi-implicit methods

Because the Newton method converges to the solution quickly if the initial guess is close to the final result, often one iteration is sufficient for oversampled methods. This is the semi-implicit method[2], which has predictable cost.

Another way to save computation is to compute the Jacobian once and use it several times in the iterations (chord method)[11].

2.1.8. Approximation of ODE by static nonlinearity and digital filters

This is not an ODE method but approximates the result. It provides a baseline to evaluate the significance of the memory in the nonlinearity. The comparison also demonstrates what is lacking when static nonlinearities are used.

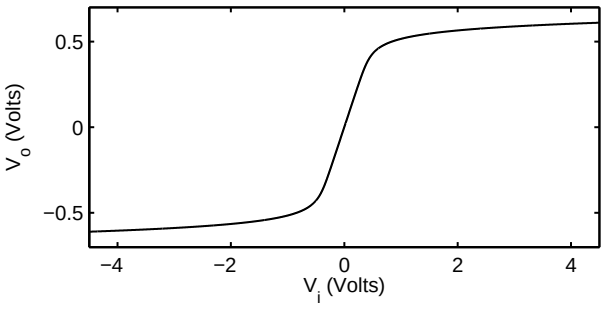


Figure 2: *Tabulated static nonlinear function*

The nonlinearity used is the DC approximation of the actual nonlinearity (Fig. 2), which can be derived from (2) by setting $C\frac{\mathrm{d}V}{\mathrm{d}t} = 0$. This is implemented using a lookup table as in [1] and is also known as waveshaping distortion. It is found that using a first-order low-pass filter before the nonlinearity with a cutoff frequency determined by the R and C of the diode limiter reduces aliasing while maintaining accurate output.

2.2. Order and Accuracy

The traditional measure of accuracy is Local Truncation Error (LTE), which is the lowest order difference between the Taylor series expansion of the solution and the result of the method. Other manifestations of error are aliasing and frequency warping. Oversampling reduces error by the order of the method. Because aliasing is more a function of the nonlinearity than of the method, it determines the minimum oversampling factor required. Frequency warping is not a concern in the audio band at these high sampling rates.

2.3. Stability

Consider the linear system described by (4) with $B = 0$. Substituting this into the Forward Euler method (5), for example, yields:

$$y_n = (I + TA)y_{n-1} + TBu_{n-1} \tag{12}$$

This is stable if $|1 + T\lambda| < 1$ for each eigenvalue λ of matrix A. The stability of an ODE method depends on the ratio between the sampling frequency and the largest eigenvalue of the system A.

2.3.1. Explicit methods

The plot of the region of stability on the $T - \lambda$ plane often forms a bounded region where the method is stable. Explicit methods result in polynomial stability conditions [6], which trace out a stability region that is a subset of the left half s-plane. Consequently, this places a limit on the largest magnitude negative eigenvalue the system may assume to assure bounded behavior.

2.3.2. Implicit methods

For implicit methods, the stability region extends to infinity in the negative half-plane, thereby placing no limit on the maximum

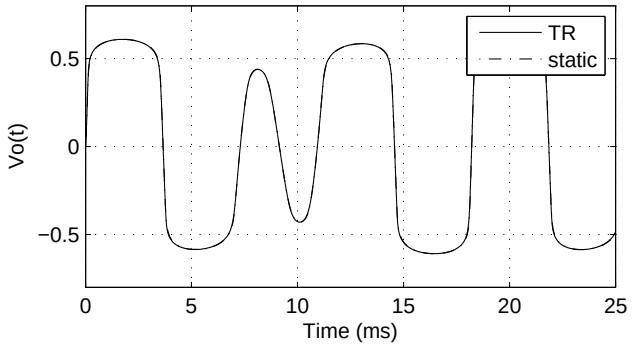


Figure 3: *Time-domain waveform for 110Hz + 155 Hz input, Trapezoidal (TR) and static approx., 8x oversampling. They are indistinguishable in the figure.*

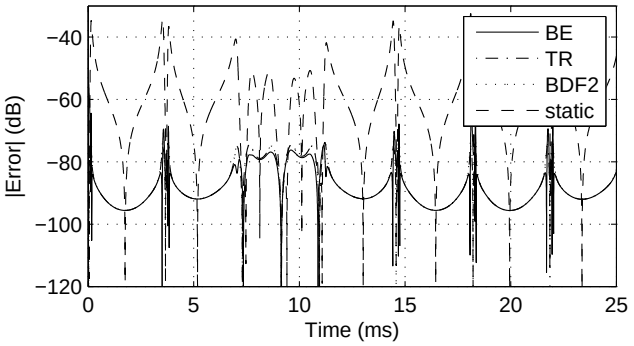


Figure 4: *Time-domain dB error for 110Hz + 155 Hz input, 8x oversampling. Backward Euler, Trapezoidal, BDF2, and static approx.*

magnitude of an eigenvalue of a system, if it is not complex, and allows a low sampling rate. For Trapezoidal, Backward Euler and BDF2, the regions encompass all of the left half-plane, so all stable systems will map to stable discrete time systems ("A-stability"). Backward Euler and BDF2 will introduce artificial damping to higher-frequency poles, while Trapezoidal Rule applies the bilinear transform to the poles and introduces no additional damping.

2.3.3. Stiff stability

For the ODEs found in circuits, it has been found in practice that implicit methods drastically reduce the sampling rate requirement relative to explicit methods, and are ultimately more efficient[10]. In the ODE literature this property is known as "stiffness." Stiffly stable solvers place no requirement on the minimum sampling rate needed to ensure a bounded solution. Instead, considerations for accuracy, in this case aliasing, govern the choice of step size.

All explicit methods are not stiffly stable [6] and they require a minimum sampling rate to operate properly. The left half plane eigenvalue of the diode clipper can be found from a small-signal linearization of the circuit. The linearized circuit is shown in Fig. 1. When V_o is large the linearized diode resistance will dominate over R, making this eigenvalue approximately

$$\lambda_{\text{clip}} \approx -C\frac{V_t}{2I_s}\text{sech}(V_o/V_t). \tag{13}$$

3. COMPARATIVE RESULTS AND DISCUSSION

3.1. Single-frequency sine

The implicit and semi-implicit methods at $8\times$ oversampling generate almost identical time-domain responses in response to a dual-tone excitation (110 and 155 Hz, 4.5 V peak), so only the Trapezoidal Rule (TR) and static approximation are shown in Fig. 3. Figs. 4-5 plot the error relative to an accurate solution computed with an oversampling factor of 32. All of the numerical methods exhibit similar error profiles with very low error. The approximation shows noticeably larger error than the numerical solvers, but it is typically less than -40 dB.

A spectral comparison better represents the audible differences. The numerical solvers all produce similar output spectra

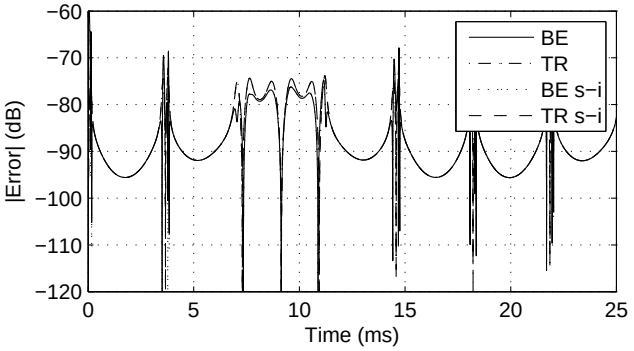


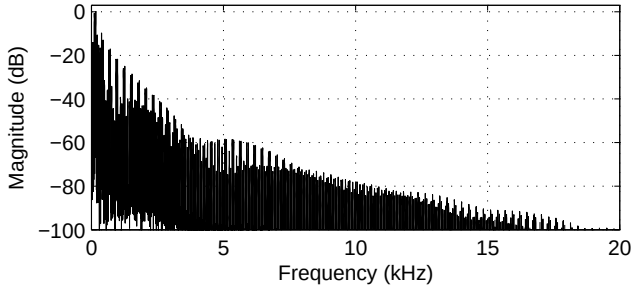
Figure 5: *Time-domain dB error for 110Hz + 155 Hz input, BE, TR and semi-implicit versions, 8x oversampling.*

and are represented in Fig. 6 by the semi-implicit trapezoidal rule, which is very accurate at eight times oversampling as indicated by the time domain error. This is compared to the static approximation, which is a close approximation that reproduces most of the major peaks and contour of the spectrum.

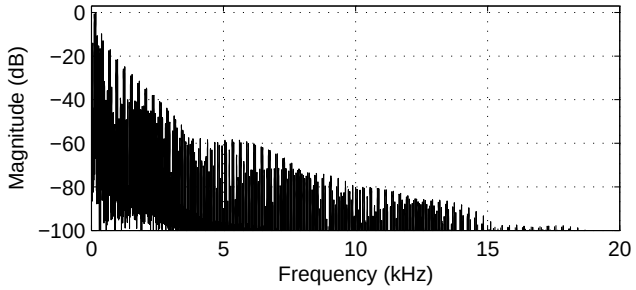
A high-level, high-frequency sine-wave excitation (4.5 V, 15001 Hz) reveals inadequacies in the semi-implicit methods, which exhibit overshoot in the time-domain plots (Fig. 7) and spurious tones in the frequency domain. The static nonlinearity produces phase error at high frequencies, although the magnitude of the fundamental is correct. The spectra of trapezoidal, semi-implicit trapezoidal, and static approximation are plotted in Fig. 8 with a reference spectrum generated by a trapezoidal rule with oversampling of 32.

3.2. Sine sweep

A high-amplitude, sinusoidal, exponential-frequency sweep from 0 to 20 kHz was processed by the methods. The oversampling factor of eight is chosen so the method is well-behaved throughout the test. The output is downsampled to 96 kHz and displayed as a log spectrogram [17]. All of the ODE methods produce almost identical output if stable, so only the spectrograms for trapezoidal rule, its semi-implicit version, and the static nonlinearity are shown in



(a) Semi-implicit trapezoidal



(b) Static nonlinearity approximation

Figure 6: *Spectra of responses for 110Hz + 155 Hz input.*

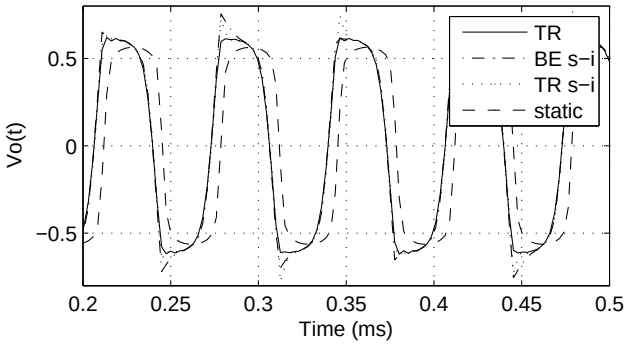


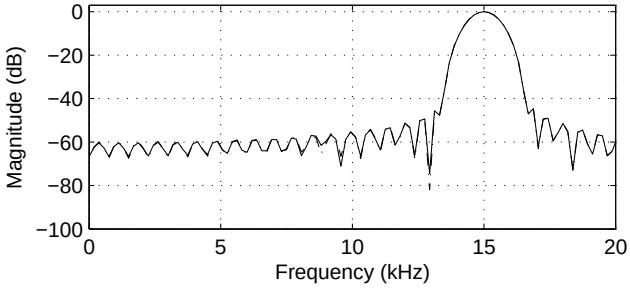
Figure 7: *Time domain waveform for 15001 Hz input, implicit and semi-implicit methods,* $8\times$ *oversampling.*

Fig. 9.

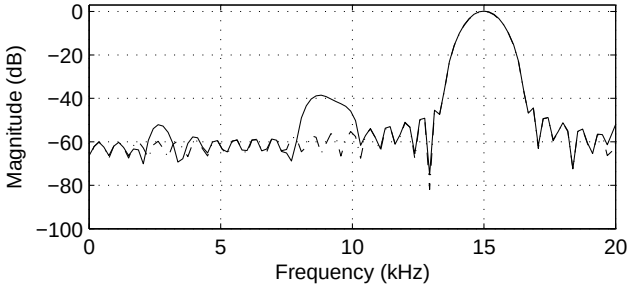
3.3. Computational cost

Because the number of iterations in an ODE solver that employs Newton's method is related to the input in a complicated way, an empirical measurement of cost is made. The number of iterations for several test signals is given in Table 1. The inputs used are an exponential sine sweep from 0 to 20 kHz, an E power chord on the low strings, and a riff with a bend.

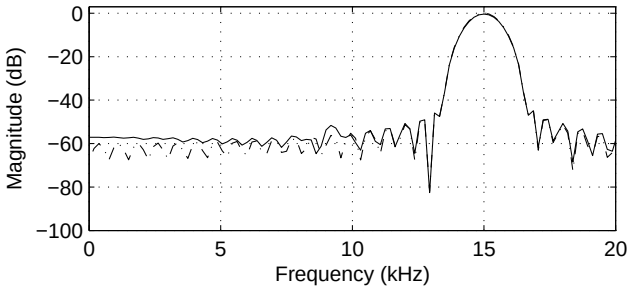
The cost per sample in terms of function calls to compute the derivative (3) or the Jacobian in the iterative methods is shown in Table 2. The cost of computing the derivative is assumed to be similar to the cost of computing the Jacobian. The cost is normalized per audio sample at the base sampling rate of 48 kHz. For iterative methods, the number of iterations n is assumed to be 1.2 as suggested by the actual guitar signals in Table 1.



(a) Solid: Trapezoidal, Dash-dot: reference



(b) Solid: Semi-implicit trapezoidal, Dash-dot: reference



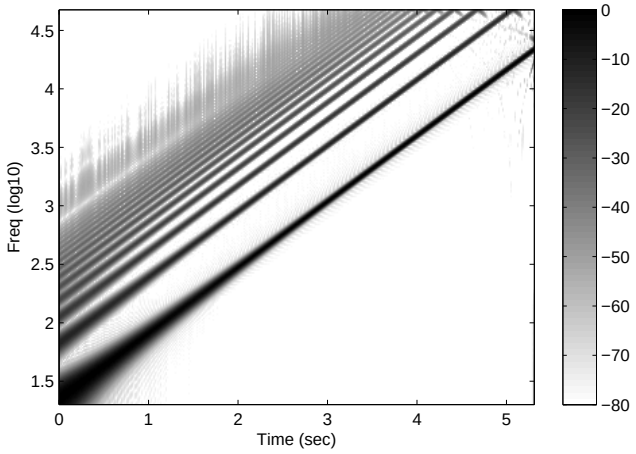
(c) Solid: Static approx., Dash-dot: reference

Figure 8: *Magnitude spectra of responses to 15001 Hz normalized to (32*$\times$*) oversampled reference.*

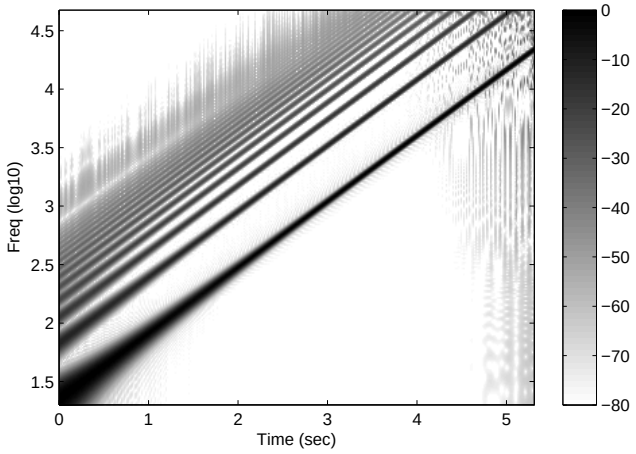
3.4. Discussion

For audio-frequency input, the differences between the methods are negligible in the audio band because the process is oversampled to reduce aliasing. The oversampling causes the various order errors to be very low in the audio band and makes the effect of frequency warping insignificant. It would seem then that a stable method of low order would be sufficient while guaranteeing bounded output.

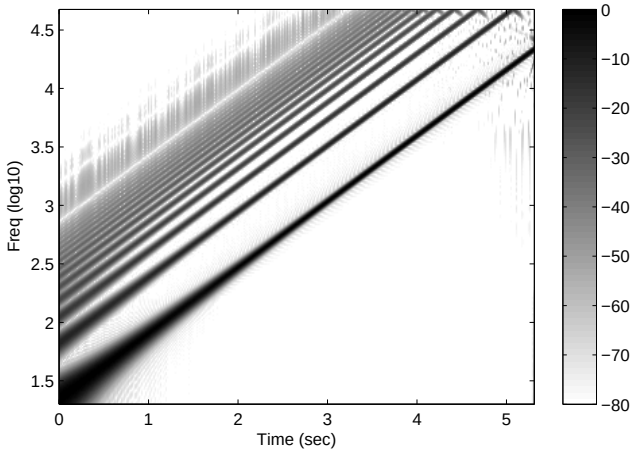
The time-domain outputs of the semi-implicit methods show significant ringing for high-frequency inputs, but this is an extreme case because high amplitudes at these frequencies are rarely encountered in practical guitar signals.



(a) Trapezoidal



(b) Semi-implicit trapezoidal



(c) Static approximation

Figure 9: *Log spectrogram of sine sweep, processed at* $8\times$ *oversampling,* $f_s = 48$ *kHz, 80-dB dynamic range.*

Input	BE	BE s-i	TR	TR s-i	BDF2
sine sweep	1.381	1	1.370	1	1.367
power chord	1.035	1	1.035	1	1.035
riff	1.1821	1	1.1814	1	1.1815

Table 1: *Number of iterations for (semi-)implicit methods: Backward Euler, semi-implicit BE, Trapezoidal, semi-implicit TR, BDF2*

Method	X	f-calls	f-calls/sample
FE	38	1	38
RK4	30	4	120
BE	8	$2n$	19.2
BE s-i	8	2	16
TR	8	$1 + 2n$	27.2
TR s-i	8	3	24
BDF2	8	$2n$	19.2
static	8	lookup	-

Table 2: *Cost comparison of methods: Oversampling* (X) *required. Calls to derivative or Jacobian function. Calls per audio sample,* $n = 1.2$ *for iterative methods.* $n =$ *# iterations; base sampling rate = 48 kHz*

The explicit methods, while simple, do not produce reliably accurate results in the frequency domain unless they are highly oversampled. The evaluation of cost validates prior findings in the circuit simulation literature that implicit or semi-implicit methods are preferred over explicit ones due to the large oversampling required for the explicit methods to be stable.

4. CONCLUSIONS

One important factor in deciding the sampling rate is the inevitable problem of bandwidth expansion caused by a nonlinearity, which may result in aliasing in sampled systems. Because the nonlinearities are strong, bandwidth is expanded by a large factor, necessitating large oversampling rates. This constraint on the sampling rate causes the different methods to be negligibly different in the audio frequency band. One can conclude that the simplest possible method that produces stable output should be used to save computational cost.

The amount of iterations taken by implicit methods depends on the frequency of the input signal. Quickly moving signals relative to the step size cause the initial estimate of the state to be far from the solution, requiring many iterations. Because the process is already highly oversampled relative to the bandwidth of realistic guitar signals to reduce aliasing, semi-implicit methods may work well. Future work could improve upon the convergence of the semi-implicit methods by limiting the overshoot for fast signals.

The static approximation of the ODE, which involves a prefilter and the DC transfer curve, is seen to be a cheap and effective approximation of the ODE, validating its widespread use in digital implementations of distortion effects. For future work, it is desirable to find other low-cost approximations of the ODE that result in better accuracy in the output spectrum.

At the very least, the application of ODE solvers to nonlinear musical effects builds insight into the problem and provides a reference point by which to evaluate more efficient approximations.

5. ACKNOWLEDGMENTS

David Yeh is supported by the Stanford, NDSEG and NSF graduate fellowships. Thanks to Andrei Vladimirescu for his advice and expertise on SPICE circuit simulation. Thanks to Stefan Bilbao for spending an afternoon and some commute time to work out ODEs and to explain numerical methods to me.

6. REFERENCES

[1] D. T. Yeh, J. Abel, and J. O. Smith, "Simplified, physically-informed models of distortion and overdrive guitar effects pedals," in *Proc. of the Int. Conf. on Digital Audio Effects (DAFx-07)*, Bordeaux, France, Sept. 10–14, 2007.

[2] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, Cambridge, MA, second edition, 1992.

[3] T. Hèlie, "On the use of Volterra series for real-time simulations of weakly nonlinear analog audio device: application to the Moog ladder filter," in *Proc. of the Int. Conf. on Digital Audio Effects (DAFx-06)*, Montreal, Quebec, Canada, Sept. 18–20, 2006, pp. 7–12.

[4] J. S. Abel and D. P. Berners, "A technique for nonlinear system measurement," in *Proc. 121st AES Convention*, San Francisco, CA, Oct. 2006.

[5] S. Bilbao, "Personal communication," Nov. 2006.

[6] J. Stoer and R. Rulirsch, *Introduction to Numerical Analysis*, Springer-Verlag, New York, third edition, 2002.

[7] W. C. Gear, *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice-Hall, Englewood Cliffs, NJ, 1971.

[8] L. F. Shampine, *Numerical Solution of Ordinary Differential Equations*, Chapman and Hall, New York, 1994.

[9] A. Vladimirescu, *The Spice Book*, Wiley, New York, 1994.

[10] W. J. McCalla, *Fundamentals of Computer-Aided Circuit Simulation*, Kluwer Academic Publishers, Boston, 1987.

[11] L. F. Shampine and M. W. Reichelt, "The MATLAB ODE Suite," *SIAM Journal on Scientific Computing*, vol. 18, pp. 1–22, 1997.

[12] M. Karjalainen and J. Pakarinen, "Wave digital simulation of a vacuum-tube amplifier," in *IEEE ICASSP 2006 Proc.*, Toulouse, France, 2006, vol. 5, pp. 153–156.

[13] M. Bosi and R. Goldberg, *Introduction to Digital Audio Coding and Standards*, Klewer Academic Publishers, 2003.

[14] S. Möller, M. Gromowski, and U. Zölzer, "A measurement technique for highly nonlinear transfer functions," in *Proc. of the Int. Conf. on Digital Audio Effects (DAFx-02)*, Hamburg, Germany, Sep. 26-28 2002, pp. 203–206.

[15] K. Meerkötter and R. Scholz, "Digital simulation of nonlinear circuits by wave digital filter principles," in *Proc. IEEE Int Symp on Circuits and Systems*, May 1989, vol. 1, pp. 720–723.

[16] A. Sarti and G. De Poli, "Toward nonlinear wave digital filters," *IEEE Trans. Signal Process.*, vol. 47, pp. 1654–1668, June 1999.

[17] J. C. Brown, "Calculation of a constant-Q spectral transform," *J. Acoust. Soc. Am*, vol. 89, pp. 425–434, 1991, Matlab code available at http://web.media.mit.edu/ brown/cqtrans.htm.

[18] D. T. Yeh and J. O. Smith, "Discretization of the '59 Fender Bassman tone stack," in *Proc. of the Int. Conf. on Digital Audio Effects (DAFx-06)*, Montreal, Quebec, Canada, Sept. 18–20, 2006, pp. 1–6.

[19] A. Huovilainen, "Nonlinear Digital Implementation of the Moog Ladder Filter," in *Proc. of the Int. Conf. on Digital Audio Effects (DAFx-04)*, Naples, Italy, Oct. 5–8, 2004.

[20] U. Zölzer, Ed., *DAFX – Digital Audio Effects*, J. Wiley & Sons, 2002.

[21] M. Karjalainen, T. Mäki-Patola, A. Kanerva, and A. Huovilainen, "Virtual air guitar," *Journal of the AES*, vol. 54, no. 10, pp. 964–980, Oct. 2006.

A GENERIC SYSTEM FOR AUDIO INDEXING: APPLICATION TO SPEECH/ MUSIC SEGMENTATION AND MUSIC GENRE RECOGNITION

Geoffroy Peeters
IRCAM - Sound Analysis/Synthesis Team, CNRS - STMS
Paris, France
peeters@ircam.fr

ABSTRACT

In this paper we present a generic system for audio indexing (classification/ segmentation) and apply it to two usual problems: speech/ music segmentation and music genre recognition. We first present some requirements for the design of a generic system. The training part of it is based on a succession of four steps: feature extraction, feature selection, feature space transform and statistical modeling. We then propose several approaches for the indexing part depending of the local/ global characteristics of the indexes to be found. In particular we propose the use of segment-statistical models. The system is then applied to two usual problems. The first one is the speech/ music segmentation of a radio stream. The application is developed in a real industrial framework using real world categories and data. The performances obtained for the pure speech/ music classes problem are good. However when considering also the non-pure categories (mixed, bed) the performances of the system drop. The second problem is the music genre recognition. Since the indexes to be found are global, "segment-statistical models" are used leading to results close to the state of the art.

1. INTRODUCTION

Automatic audio indexing has become a major concern today. Given the increasing amount of audio indexing applications (sound recognition, music genre/ mood recognition, singer type recognition, speaker recognition, speech/ music segmentation...) many different applications have been, are and will be developed. However most of these applications rely on the same underlying concepts: extract a set of time-frame feature vectors, train a statistical model using hand-labeled data in order to create a "classifier" and finally use this classifier to label unknown data. Because of that, developing a unique generic and modular indexing system is attractive.

In the ongoing French national project "Ecoute", two of these indexing applications are to be developed: a speech/ music segmentation and a music genre recognition system. It has therefore been decided to develop this generic audio indexing system and apply it to the two problems. The goal of this paper is to present this generic indexing system and detail its application to the two problems.

Several generic systems have been proposed so far. For example, the Waikato University WEKA [1] system is a generic machine learning system written in Java. However its direct applicability to the audio case is not obvious (no feature extraction, no consideration of time information). The Sony EDS [2] system performs both feature extraction and machine learning but is heavy in computation time. The McGill University jAudio[3] + ACE[4], Tzanetakis' Marsyas[5], or IMIRSEL's M2K [6] systems all seem promising solutions but it still need to be proven that they provide large performances for specific applications.

Our generic system is based on a system we previously developed for a task of instrumental sound recognition[7]. For this task the system showed very good performances. The training stage of the system is based on a succession of four steps: feature extraction, feature selection, feature space transform and statistical modeling. The system has been modified and extended to make it generic and modular. The requirements for the design of such a generic system are presented in part 2.1. The system we have developed is presented in part 2.2. We then present the results of applying it to the two considered problems: speech/ music segmentation (part 3.1) and music genre recognition (part 3.2).

2. GENERIC AUDIO INDEXING SYSTEM

2.1. Requirements for a generic system

The two main actions the system must perform are training and indexing. "Training" denotes the stage in which a classification model is learned from hand-labeled data. "Indexing" denotes the stage in which this classification model is used to label or segment unknown data. The two actions must be clearly separated since they are not used by the same people.

Training consists in extracting features from a set of audio files (or a database) and finding a mapping between the characteristics of the features and hand-annotated labels of the audio files. An audio file can have a unique label (for example a "music genre" label describes a whole music track file) or a succession of labels over time (a 24h radio program file is described by a succession of labels over time: speech, music, jingles...). These labels define the problem to be solved. The set of files and the corresponding labels must be easy to define and modify by the user.

The performances of the system depend strongly on the features used. Each problem may require different features. Therefore, in a generic system, changing the feature extraction stage must be easy. Conversely the system must be able to choose by itself the appropriate subset of audio features in order to solve a specific problem.

The performances of the system also depend strongly on the choice of the model used to represent the mapping between the features and the labels (the classification model). SVM is known to perform the best but is limited to two classes problems, KNN also perform very well but is limited by the size and dimensionality of the training set. This part of the system must also be easily parametrizable.

Part of the training consists in testing the performances of the trained model. Several evaluation methods can be used for that:

cross-database validation, N-folds cross validation or Leave-One-Out validation.

The system we have developed takes the previous requirements into account. The global flowchart of the system is presented in Fig. 1. We describe it in the following part.

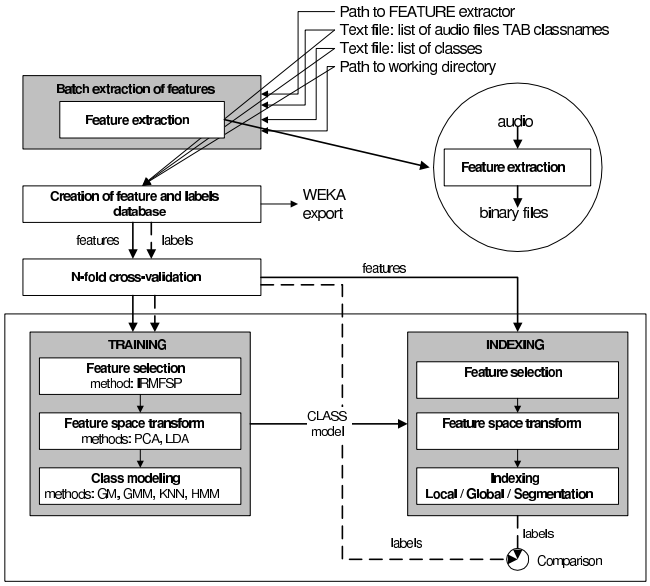


Figure 1: *Flowchart of our generic indexing system.*

2.2. Description of the system

2.2.1. Describing a new indexing problem

In order to make the description of a new indexing problem easy for the user, we have chosen to use a simple set of text files.

List of audio files and annotations: The first text file contains a list of audio files that will be used by the system to learn the characteristics of the classes. Each row of the text file contains the path to an audio file followed by the name of the corresponding annotated class. The user also has the possibility to replace the name of the annotated class by the path to a Wavesurfer[8] .lab file which allows to annotate, for the same audio file, a succession of classes over time.

List of classes and mappings: The second file contains the list of the name of the classes that will be considered for the training. This list can be a subset of the classes used for the annotation of the audio files (we call the later "annotated classes"). In this case, only the files (or the temporal segments) corresponding to this subset will be considered during the training. This file can also perform a mapping between the annotated classes and new class names. This allows mapping annotated class names between various databases. Several annotated classes can also be mapped to the same new class. This allows creating hierarchy among classes. For example combining the annotated "talk-voice" and "ads-voice" classes (see part 3.1) into a unique trained "speech" classes is very easy with our system.

Extractor: Finally, the last input to the system is the path to a "feature extractor". A "feature extractor" is a program that takes as input the path to an audio file and output the features values over time in a binary file. The format of the output file is self-defined in order to 1) make it usable without the knowledge of the feature extractor, 2) gives the necessary information in order to guarantee that all the feature files used by the system are compatible. In order to do that, each file contains the feature values, feature names, an identifier to the used feature extractor, the parameters of it and its version.

2.2.2. Training the system

In order to train the system, we first extract all the features of the audio files to be used as examples to learn the classes. This is done in a batch process using all the files defined in the list of audio files and using the defined "feature extraction" program. The results of this is a set of binary feature files.

A database is then created containing all the feature and class values. For this, the system reads all the binary files containing the features over time, reads the class definitions and perform the mapping between the features and the classes. At this stage, the user can export all the data in the Weka[1] format in order to perform external statistical analysis.

The training of the system then starts. It is a succession of three stages.

Feature selection: The first stage of the system selects among all extracted features the ones that are the most useful to describe the classes. The algorithm currently used is the Inertia Ratio Maximization with Feature Space Projection (IRMFSP) we proposed in [7].

The IRMFSP algorithm measures for each individual feature (we consider a multi-dimensional feature, such as the MFCC, as a set of scalar features) the inertia ratio (also named Fisher discriminant) knowing the feature values and their class belonging. The algorithm then selects the feature with the largest inertia ratio (the most discriminative feature). It then applies an orthogonalization process by projecting the whole feature space on the selected feature. This process guarantees that the remaining features are orthogonal to the selected feature (i.e. no more correlated). The process is then repeated in order to select the next features.

Feature space transform: The second stage transform the feature space in order to reduce its dimensionality while improving class representation. Currently two transforms are used: -the PCA which reduces the dimensionality of the feature space while preserving most of the variance of the data, -the LDA which reduces the dimensionality while maximizing the class separation of the data.

Class modeling: Finally, the third stage performs the statistical modeling of the problem. The following models are currently available: multi-dimensional Gaussian modeling, Gaussian mixture modeling, K-Nearest Neighbors, hidden Markov models, various unsupervised clustering algorithms, and histogram learning.

The output of the training is a "CLASS model" file, which stores all the parameters of the training and the references to the extractor to be used. This file can then be used for the indexing of unknown files.

2.2.3. Indexing

The current system can process the two following types of indexing:

Local indexing means that various labels are assigned over the file duration. In this case, each time frame (feature vector) is processed separately and classified separately. Smoothing techniques over time can be applied by computing short-term histogram of class belonging or by applying median filtering. Class changes over time can then be used to perform segmentation and assign a label to each segment. The local indexing will be used in part 3.1 for the speech/ music segmentation problem.

Global indexing means that a single global label is assigned to the whole file (or segment) duration. This is the case when

1. the feature vector is timeless, i.e. it describes directly the whole file (or segment) duration (this was the case in our instrumental sound classifier[7]),
2. when a global decision is taken from a succession of instantaneous features. This is the case when using hidden Markov modeling or when using the methods proposed below.

The results of the indexing process is output in a simple text file (Wavesurfer format).

2.2.4. Global indexing methods

In a standard classification system, each feature-vector at each time frame $f(t)$ is considered independently as an observation of a specific class c_i. The training and indexing are therefore performed directly on a frame basis. We call this model a "frame-statistical model".

When all the frames of a given file (segment) are supposed to belong to the same class, one can benefits from this knowledge by performing a "vote" among the frame-classes. This allows improving the accuracy of the classification by reducing the effect of local (frame-class) misclassification. We present below the "cumulated histogram" and the "cumulated probability" method.

In some cases (as for example the music genre problem of part 3.2), a segment (file) belongs as a whole to a class-to-be-found, but its individual frames do not necessarily belong to the class-to-be-found (in the case of music genre, a given time frame of a rock song can be very close acoustically to a given time frame of a blues song, therefore the various time frames of a given track could belong to various classes while the whole track belongs to only one class). The class-to-be-found is rather defined by a specific distribution or succession over time of frame-classes. The "segment-statistical model" presented below allows to take this into account.

"Cumulated histogram": The decision about the global class of a file/ segment is made by choosing the class with the largest number of occurrences among the frames. For this, each frame of the file/ segment is first classified separately: $i(t) = \arg\max_i p(c_i|f(t))$. The histogram $h(i)$ of class belonging $i(t)$ over all the frames is then computed (the bins of the histogram corresponds to the various classes c_i). The class corresponding to the maximum of the histogram $h(i)$ is finally chosen as the global class. In the following we call this method "cumulated histogram".

"Cumulated probability": Another possibility, is to use the frame probabilities $p(c_i|f(t))$, cumulate them over all the frames belonging to the file/ segment ($p(c_i) = \frac{1}{T}\sum_t p(c_i|f(t))$ and choose the class i with the highest cumulated probability ($i = \arg\max_i p(c_i)$). We call this method "cumulated probability".

"Segment-statistical model": In this paper, we propose to learn the characteristics of the "cumulated probability" and use the corresponding statistical models to perform the classification. We note s a specific segment (file) and $p_s(c_i)$ its cumulated probability. We note S_i the set of segments (files) of the training set belonging to a specific class i. For each class i, we compute the set of "cumulated probabilities" $p_{s\in S_i}(c_i)$. For a specific class i, we then model the behaviors of the bins c_i over all the $s \in S_i$. We call this model a "segment-statistical model" and note it $\hat{p}_i(c_i)$. In order to index an unknown segment/ file, we first compute its "cumulated probability" $p_s(c_i)$ and classify it using the trained "segment-statistical models" $\hat{p}_i(c_i)$.

Two statistical models have been considered:

1. The first one uses a Gaussian modeling of the bins of c_i. For each class i, we compute the mean and standard deviation of each bin c_i over all the $s \in S_i$.
2. The second model uses only the above-mentioned mean values. In this case, the indexing is performed by choosing the class i corresponding to the model $\hat{p}_i(c_i)$ with the largest cosine distance with the "cumulated probability" $p_s(c_i)$ of the unknown segment (file) s.

2.2.5. Validation

The current system can perform two types of validation.

Cross-database validation: one database is used for training the system, another one is used to evaluate the performances of it.

N-folds cross validation: the set of data is divided into N-folds: N-1 of them are used for training the system, the remaining one is used to evaluate the performances of it. In this case, special care must be taken in order to guarantee independence between the folds used for training and the one used for evaluation. In our system, we use the folder information of the files to detect dependencies. A specific case of N-folds cross validation is the Leave-One-Out validation in which N equal the number of data.

In part 3.1, we will used both validation methods, in part 3.2, only cross-database validation will be used.

2.2.6. Features

So far, two feature extractors have been developed.

Dedicated audio features: The first extractor we have developed is dedicated to the problem of instrument sound recognition and is described in details in [9]. In this extractor, the assumption is made that the audio file contains a single instrument note. Therefore the extraction of high-level concepts (such as attack time or fundamental frequency of a note) is *-feasible* (i.e. can be done considering current signal processing capabilities) and -*meaningful* (i.e. has a meaning for the given signal).

Generic audio features: In the case of generic audio (music, radio stream...) the extraction of such concepts would be *-difficult* (requiring either temporal segmentation or source separation) and *-meaningless* (considering that a 24h radio program or a music track has more than one attack time or release time). Therefore the second extractor we have developed only contains the subset of features that do not rely on any time model (such as the temporal organization assumption necessary to derive the attack time) or signal model (such as the harmonic sinusoidal model necessary to derive the fundamental frequency). It extracts instantaneous features such as MFCC and Spectral Flatness Measure.

2.2.7. Temporal modeling of features

Instantaneous features are usually extracted using a 40ms window with a hop size set to 20ms. This can lead to a very large amount of data for the training: 4 millions feature vectors for a 24 hours radio program file. In order to decrease the amount of data, "temporal modeling" of the feature vectors can be performed.

"Temporal modeling" means modeling the evolution over time of each feature using frame analysis. The length of the sliding window is typically chosen between 500ms to 2s and the modeling is performed over each window. The current system can perform the following type of "temporal modeling": statistical measures (mean, variance values over each window), histogram of cluster belonging, spectral decomposition of feature evolution [10] and sub-band grouping of this spectral evolution [11] [12].

3. TWO APPLICATIONS OF THE INDEXING SYSTEM

3.1. Speech/ music segmentation

The first application we present is a tool for automatic segmentation of radio streams. This tool is developed in coordination with a company that produces managing and archiving softwares for radio stations. The categories to be indexed as well as the radio corpuses are directly defined and provided from their clients and are thus real world categories and data.

Speech/ music segmentation has been the subject of a large amount of research in the last two decades. The front-end of most systems starts by extracting low-level features such as the zero-crossing rate, 4 Hz energy modulation, spectral moments, MFCC, entropy... Each class is then modeled using instance-based classifier (KNN), Bayesian classifier (Gaussian mixture model), Support Vector Machine... The field is well established and has dedicated evaluation protocols such as DARPA in the USA or ESTER [13] in France. We refer the reader to [14], [15], [16], [17], [18], [19] for major or recent publications in this field.

The goal of this part is twofold: first we want to test the applicability of our system for a task of speech/ music segmentation, secondly we want to test such a system in a real industrial framework.

3.1.1. Considered categories

Two sets of categories are to be found. The first set corresponds to *acoustical categories*: music, voice, mix and bed.

- "Mix" denotes segments where music and speech exist but do not overlap continuously over time; they rather succeed each other over time.
- "Bed" denotes segments where speech and music overlap regularly over time; a typical example of it is the introduction of news on the radio.

The second set of categories corresponds to *industry categories*, i.e. the categories used by radio programmers to annotate their programs: music, talk, ads and jingles. Obviously the categories "talk" and "ads" can be composed of voice, mix or bed. The category "jingle" can also be composed of any of the previous acoustical categories. We do not detail the "jingle" part of the system in this paper since it is processed by a dedicated audio finger-print system which allows to identify them. The correspondence between the industry and the acoustical categories is indicated into Tab. 1.

3.1.2. Corpus

The corpuses used for the development and testing of the system are the following:

Corpus RadioFrance: the speech part is composed of a subset of the MPEG-7 corpus made of recordings of Radio-France radio station in July 98, the music part is composed of two subsets: the ISMIR2004 "song excerpts" test set and a private music genre database.

Corpus UK: consists of 24h of recording of a major commercial radio group in the UK. This station has a high rate of audio compression, includes many ads, jingles, talks and music.

Corpus SUD: consists of 24h of recording of a regional radio station in France.

Each corpus has been annotated into the above mentioned categories. However, the RadioFrance corpus is only annotated in the categories music-music and speech-clean (equivalent to talk-voice and ads-voice). The annotations are in the Wavesurfer format. The distribution of the corpuses is indicated into Tab. 2. For each category we indicate its percentage (%) and its duration in minutes (m). As one can see, all three corpuses are highly unbalanced in favor of the music category.

3.1.3. System configuration

Features: For each corpus, we have extracted the following set of instantaneous audio features:

- 13 Mel-Frequency-Cepstral-Coefficients (using 40 triangularly-shaped Mel bands, and keeping the DC component),
- Delta-MFCC,
- Delta-Delta-MFCC,
- 4 Spectral-Flatness-Measure coefficients (the 4 rectangularly-shaped frequency bands are [250, 500], [500, 1000], [1000, 2000] and [2000, 4000] Hz),
- Delta-SFM,
- Delta-Delta-SFM.

The signal is first converted to mono and down-sampled to 11KHz. The frame analysis was performed using a 40ms Blackman window, the hop size was 20ms. We then apply temporal modeling to the 50Hz feature vector signal using the mean and variance values over a 2s window with a hop size of 1s.

Classifier: Various configurations of the classifier have been tested (variation of the number of selected features, choice of the statistical model, variation of the number of mixtures in the GMM...). The best results were obtained with the following configuration:

- Feature selection: IRMFSP algorithm using the first 40 selected features,
- Feature space transform: Linear Discriminant Analysis,
- Class modeling: GMM with 20 mixtures and full-covariance matrix. Since the corpus is highly unbalanced we did not use the prior probabilities in the Bayes formulation.

Industry / Acoustical	Music	Jingle	Voice	Mix	Bed
Music	music-music				
Jingle		jingle-jingle			
Talk			talk-voice	talk-mix	talk-bed
Ads			ads-voice	ads-mix	ads-bed

Table 1: *Correspondence between industry and acoustical categories for speech/music segmentation*

Corpus name		RadioFrance		UK	SUD
Description		french speaking		english speakin	french speaking
Total duration		622m		1375m	1333m
Classes:	music-music	74% (460m)	music-music	57% (788m)	70% (945m)
	speech-clean	26% (162m)	talk-voice	16% (222m)	23% (312m)
			talk-mix	8% (111m)	1% (13m)
			talk-bed	3% (41m)	1% (10m)
			ads-voice	4% (51%)	1% (10m)
			ads-mix	6 (89m)	3% (35m)
			ads-bed	5% (70m)	1% (8m)

Table 2: *Distribution of the three corpuses for speech/music segmentation*

		Found						
		'music-music'	'talk-voice'	'talk-mix'	'talk-bed'	'ads-voice'	'ads-mix'	'ads-bed'
Real	'music-music'	79,4	0,5	1,7	2,9	0,9	8,5	6,1
	'talk-voice'	0,5	71,8	8,1	5,0	12,4	1,3	0,8
	'talk-mix'	2,6	8,3	42,6	22,2	6,3	9,1	8,9
	'talk-bed'	4,1	3,9	34,9	39,8	5,3	6,4	5,6
	'ads-voice'	1,1	10,0	5,8	3,1	66,2	9,2	4,4
	'ads-mix'	12,1	2,3	9,4	5,8	10,4	41,7	18,3
	'ads-bed'	6,8	0,8	6,0	5,3	6,1	14,4	60,6

57,5

Table 3: *Ten-fold cross-validation confusion matrix of the speech/music system for the 7 categories problem using the UK corpus*

3.1.4. Results

7 classes problem: We first present the results obtained when considering blindly the 7 classes problem (blindly means that we do not take into account the fact that some classes are in fact acoustically equivalent): music-music, talk-voice, talk-mix, talk-bed, ads-voice, ads-mix and ads-bed. The results obtained using a ten-fold cross-validation method for the UK corpus (the most difficult) are indicated in Tab. 3. The average Recall (average over the classes) is $\overline{R} = 57.5\%$ (the random Recall for 7 classes would be $\overline{R} = 14.28\%$). Music-music and talk-voice are recognized at $R = 79.4\%$ and $R = 71.8\%$ respectively. The largest confusions occur with the non-pure categories (mix and bed) and when trying to distinguish talk from ads. The category talk-voice is mainly confused with ads-voice/ talk-mix/ talk-bed, the category talk-mix with talk-bed/ ads-mix/ ads-bed, the category ads-voice with talk-voice...

Mean Recall (Mean F-Measure) Music Recall - Speech Recall		Evaluation: Radio-France	Evaluation: UK	Evaluation: SUD
Training	Radio-France		**86,5 (87,9)** 99 - 73,9	**95,2 (96,4)** 90,9 - 99,5
	UK	**92,1 (57,6)** 84,3 - 99,9		**89,4 (92,7)** 79,1 - 99,8
	SUD	**95 (78,1)** 96,9 - 93,2	**90,2 (91,3)** 99,1 - 81,3	

Table 4: *Cross-database evaluation of the speech/music system for the 2 categories system using the three corpuses*

2 classes problem: We now only consider the pure categories and merge the acoustically equivalent categories. This leads to two classes: music-music and a category merging the categories talk-voice and ads-voice, which we call speech. For the UK corpus, using a ten-fold cross-validation method, the mean Recall is $\overline{R} = 95.6\%$ ($R_{music} = 96.7\%$ and $R_{speech} = 94.4\%$), for the Radio-France corpus it is $R_{music} = 96.48\%$ and $R_{speech} = 96\%$, for the SUD corpus it is slightly lower: $R_{music} = 95.8\%$ and $R_{speech} = 92.1\%$. Whatever the considered number of classes or the considered corpus, music tends to be more easily recognized than speech.

Cross-database validation: We now want to test the generability of the trained classification model. In particular, we want to test if the system has learned the general characteristics of music and speech or the specific characteristics of music and speech as played on the specific radio station used for training. In order to test this, we perform a cross-validation over the three corpuses: one corpus is used for training, the two remaining ones for evaluation. The results are indicated in Tab. 4. Each cells report the mean (over the classes) Recall $\overline{R}$, mean F-measure $\overline{F}$, and the music and speech Recalls. In the following, we note $R^{x \to y}$ the Recall obtained when training the model on the corpus x and using it to classify corpus y.

The best result is obtained when training the model using the RadioFrance corpus and applying it for the indexing of the SUD corpus: $\overline{R^{RF \to SUD}} = 95.2\%$; the second best result when using SUD to classify RadioFrance: $\overline{R^{SUD \to RF}} = 95\%$. The worst results are obtained when using RadioFrance to classify UK or UK to classify SUD. RadioFrance and SUD seem very close acoustically while UK seems very different. The assumption that the difference comes from the language of the corpuses (French/ English) is contradicted by the individual class Recalls. Actually, using UK to train the speech model and applying it to the RadioFrance or SUD corpuses leads to the highest Recalls: $R_{speech}^{UK \to RF} = 99.9\%$ and $R_{speech}^{UK \to SUD} = 99.8\%$ respectively. The difference between the corpuses seems to come mainly from the music part: $R_{music}^{UK \to RF} = 84.3\%$ and $R_{music}^{UK \to SUD} = 79.1\%$. The music of RadioFrance or SUD tends to be recognized as the speech learned from UK. Also the speech of UK tends to be recognized as the music of RadioFrance or SUD ($R_{speech}^{RF \to UK} = 73.9\%$ and $R_{speech}^{SUD \to UK} =$

81.3%). The music model is better trained using the SUD corpus: $R_{music}^{SUD \rightarrow RF} = 96.9\%$ and $R_{music}^{SUD \rightarrow UK} = 99.1\%$.

Comments on the Precision and F-measure: The values of the F-measure, or the Precision factor, must be analyzed with care since they strongly depends on the distribution of the test set which is highly unbalanced in our case. For example in the case of RadioFrance classified by UK, we get 99.9% Recall for the class "speech", but its Precision is only 13.4%. This looks like a large part of "music" has been classified as "speech". In fact this part is small in comparison to the number of "music" data: only 15.6% of the music data have been classified as speech. But since the total number of music data (m=48382 data) is much larger than the total number of speech data (s=1175), even 15.6% (0.156*m=7581) makes the Precision drops drastically (the Precision is computed as 0.999*s/(0.156*m+0.999*s).

Conclusion: Considering that no specific modifications of our system have been made for the specific task of speech/ music indexing, the results obtained for the two-classes problem are very encouraging. The choice of the training set seems however to be important for the generalization of the system and different corpus may be required for training the music and speech models. However, the application of our system for the seven-classes problem (including the non-pure categories "mix" and "bed") requires further development. In this case, the use of generic audio features (as used in our experiment) does not allow distinguishing efficiently the non-pure classes.

3.2. Music genre recognition

The second application we present is a tool for the automatic recognition of music genre. Although music genre categories have been showed to be fuzzy or hill-defined [20], their automatic estimation is a usual step in the understanding of the acoustical characteristics underlying music similarity. For this reason, it has been the subject of many contributions in recent years. Moreover dedicated frameworks, [21] or MIREX [22], are devoted to its evaluation which allows the comparison of newly developed algorithms to state-of-the-art algorithms. In opposition to speech/ music front-ends, two main categories of systems exist in the case of music genre recognition. The first one learned the classes directly from low-level features [23] [24] (MFCC, Spectral Contrast, Loudness, Roughness...). The second one learned the classes from high-level features [25] (tempo, beat histogram, chroma, pitch contours). Our system belongs to the first category since it uses the same set of low-level features as our speech/ music segmentation system. We refer the reader to [26] for a recent overview of the music genre topic.

3.2.1. Corpus and categories

For the evaluation of the performances of our system, we have used the test sets from the ISMIR2004 music genre contest [21]. It should be noted that we had only access to the training and development set, not to the evaluation one. Training of our system is done on the training set and the performances are given on the development set. The distribution of both sets are indicated in Tab. 5. As for the speech/ music corpuses, the corpus is here also very unbalanced in favor of the classical music category. The definition of the classes is also controversial: jazz and blues are merged into a single class, the "world music" class contains many different types of music.

3.2.2. System configuration

Features: The same set of features as for the speech/ music segmentation system has been used. However the modeling length was set to 4s (instead of 2s) and the hop size to 2s (instead of 1s).

Classifier: Various configurations of the classifier have been tested and the best results were obtained with the following configuration:

- Feature selection: no
- Feature space transform: Linear Discriminant Analysis,
- Class modeling: GMM with 5 mixtures and full-covariance matrix. Since the corpus is highly unbalanced we did not used the prior probabilities in the Bayes formulation.

3.2.3. Results

Since we know that all the frames of a given file belong to the same music track and should therefore have the same music genre class, we use the global indexing methods proposed in part 2.2.4. We compare the three global indexing methods (cumulated histogram, cumulated probability, segment-statistical model) to the frame-based decision method. For the "segment-statistical model" method, we only present the results obtained using the cosine-distance method (using only the mean of the bins c_i) since it leads to the highest results. However, we indicate in Fig. 2 both the mean and standard-deviation of the bins c_i for the 6 classes of our training set.

Music Genre	Classical	Jazz / Blues	World	Electronic	Metal / Punk	Rock / Pop	Total
Training set	320	26	106	115	45	101	713
Development set	320	26	122	114	45	102	729

Table 5: *Description of the training and development corpus for music genre recognition*

		Found					
		classical	electronic	jazz_blues	metal_punk	rock_pop	world
Real	classical	90,6	0,0	0,3	0,0	0,0	9,1
	electronic	1,8	73,7	0,9	2,6	9,6	11,4
	jazz_blues	0,0	0,0	96,2	0,0	3,8	0,0
	metal_punk	0,0	0,0	0,0	84,4	15,6	0,0
	rock_pop	0,0	4,9	2,9	16,7	67,6	7,8
	world	16,4	4,9	4,9	0,8	13,1	59,8

78,7

Table 6: *Confusion matrix of the music genre system for the 6 categories using the 'segment-statistical models".*

The classification on a frame-basis (87039 frames have to be classified) gives a mean Recall of $\overline{R} = 62.2\%$ ($\pm$ 14.3% variation among the classes). The "cumulated histogram" method (729 tracks have to be classified) gives a mean Recall of $\overline{R} = 76.2\%$ ($\pm$

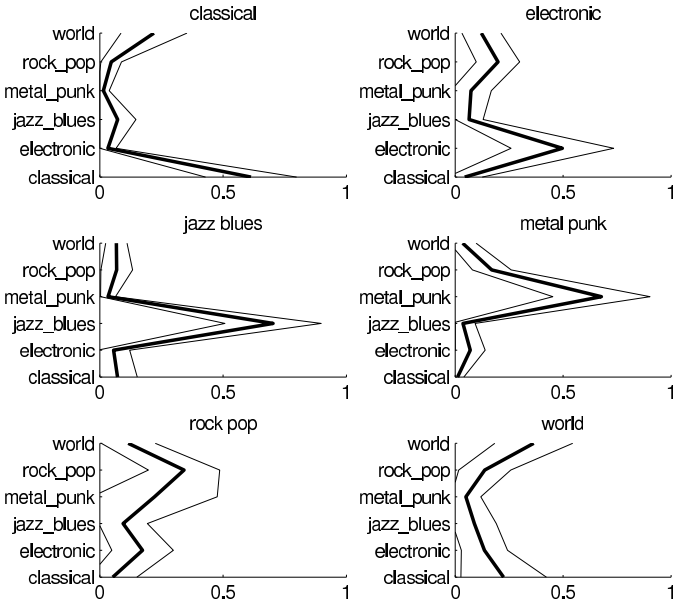


Figure 2: *Trained "segment-statistical models" for the 6 music genres: mean values (thick lines), ± standard deviation (thin lines).*

18.9%). The "cumulated probability" method gives a mean Recall of $\overline{R} = 77.4\%$ ($\pm$ 16.8%) The best method is the "segment-statistical model" with a mean Recall of $\overline{R} = 78.7\%$ ($\pm$ 14%).

In Fig. 6, we indicate the confusion matrix obtained using the "segment-statistical models". The largest confusions occur between classical and world (some world-music tracks use classical instruments), metal-punk and pop-rock (some rock songs are close acoustically to metal songs), electronic and pop-rock. It is difficult to further comment on these confusions considering the spread of the acoustical content of the categories. However, it seems clear that using only timbre-related features (such as the MFCCs and SFMs) do not allow distinguishing high-level concepts such as music genre.

Conclusion: Considering again the fact that no specific modifications of our system have been made for the specific task of music genre recognition, the results obtained are very encouraging. We get a mean Recall of 78.7%. In comparison, the results obtained by the 5 participants of the ISMIR2004 music genre contest[21] were $\overline{R} = 78.78\%, 67.22\%, 58.60\%, 55.70\%, 51.48\%$. It is important to note however that we present results for the development set while the results presented in [21] were for the evaluation set.

4. CONCLUSION AND FUTURE WORKS

In this paper we presented a generic system for audio indexing (classification/ segmentation). The system aims to be easy to use and applicable to a large range of indexing problems. We tested this by applying it to two usual problems: speech/ music segmentation of radio stream (in a real industrial framework) and music genre recognition. For this, a set of generic low-level audio features (MFCC and SFM) was used.

For the speech/ music segmentation problem, and when considering only the pure categories speech/ music, the performances of our system were good. We also showed that the system could be generalizable across datasets: a model trained on a specific radio channel could be used to index other radio channels. However when taking into account the non-pure categories ("mix" and "bed"), the performances of our system dropped.

For the music genre recognition problem, since the indexes to be found are global, we used the proposed "segment-statistical-models" leading to results close to the state of the art.

The main goal of this paper was to show the effectiveness of our generic generic system to solve quickly a specific problem. Considering the fact that we have used the same system for both problems, the results obtained are encouraging.

However the features used were very generic and therefore do not allow to represent precisely the characteristics of some classes. This was the case for the non-pure categories in speech/ music (describing these categories would involve having the possibility to observe separately the various parts of the spectrum). This was also the case for differentiating some music genres (differentiating them would require higher level musical features such as rhythm patterns, chord succession...). Future works will therefore concentrate on extending the set of audio features on which the feature selection is performed.

Another current limitation of our system comes from the unbalancing of training sets (one class is more represented than the others). In fact, real life training sets are often unbalanced. Future works will therefore concentrate in adapting our training algorithms (feature selection, feature space transforms) to this.

5. ACKNOWLEDGEMENTS

Part of this work was conducted in the context of the French RIAM project "Ecoute" (http://projet-ecoute.ircam.fr/) . Many thanks to D. Bachut for the fruitful discussions and D. Tardieu for paper corrections.

6. REFERENCES

[1] E. Frank, L. Trigg, M. Hall, and R. Kirkby, "Weka: Waikato environment for knowledge analysis," 1999-2000.

[2] F. Pachet and A. Zils, "Automatic extraction of music descriptors from acoustic signals," in *Proc. of ISMIR*, Barcelona, Spain, 2004.

[3] D. McEnnis, C. McKay, I. Fujinaga, and P. Depalle, "jaudio: A feature extraction library," in *Proc. of ISMIR*, London, UK, 2005.

[4] C. McKay, R. Fiebrink, D. McEnnis, B. Li, and I. Fujinaga, "Ace: A framework for optimizing music classification," in *Proc. of ISMIR*, London, UK, 2005.

[5] G. Tzanetakis and P. Cook, "Marsyas: a framework for audio analysis," *Organised Sound*, vol. 4, no. 3, 1999.

[6] (International Music Information Retrieval Systems Evaluation Laboratory) IMIRSEL, "Introducing m2k and d2k," in *Proc. of ISMIR*, Barcelona, Spain, 2004.

[7] G. Peeters, "Automatic classification of large musical instrument databases using hierarchical classifiers with inertia ratio maximization," in *Proc. of AES 115th Convention*, New York, USA, 2003.

[8] K. Sjolander and J. Beskow, "Wavesurfer - an open source speech tool," in *ICSLP*, 2000.

[9] G. Peeters, "A large set of audio features for sound description (similarity and classification) in the cuidado project.," Cuidado i.s.t. report, IRCAM, 2004.

[10] G. Peeters, A. Laburthe, and X. Rodet, "Toward automatic music audio summary generation from signal analysis," in *Proc. of ISMIR*, Paris, France, 2002, pp. 94–100.

[11] B. Whitman and D. Ellis, "Automatic record reviews," in *Proc. of ISMIR*, Barcelona, Spain, 2004.

[12] M. McKinney and J. Breebaart, "Features for audio and music classification," in *Proc. of ISMIR*, Baltimore, US, 2003.

[13] S. Galliano, E. Geoffrois, D. Mostefa, K. Choukri, J.-F. Bonastre, and G. Gravier, "The ester phase ii evaluation campaign for the rich transcription of french broadcast news," in *European Conf. on Speech Communication and Technology*, 2005.

[14] E. Scheirer and M. Slaney, "Construction and evaluation of a robust multifeature speech/music discriminator," in *Proc. of IEEE ICASSP*, Munich, Germany, 1997.

[15] J. Saunders, "Real-time discrimination of broadcast speech/music," in *Proc. of IEEE ICASSP*, 1996.

[16] M. Carey, E. Paris, and H. Lloyd-Thomas, "A comparison of features for speech, music discrimination," in *Proc. of IEEE ICASSP*, Phoenix, AZ, USA, 1999.

[17] H. Harb and L. Chen, "Robust speech music discrimination using spectrum's first order statistics and neural networks," in *ISPAA*, Paris, France, 2003, pp. 125–128.

[18] J. Pinquier and R. André-Obrecht, "Audio indexing: Primary components retrieval - robust classification in audio documents. dans : Multimedia tools and applications," *Multimedia Tools and Applications*, vol. 30, no. 3, pp. 313–330, 2006.

[19] G. Richard, M. Ramona, and S. Essid, "Combined supervised and unsupervised approaches for automatic segmentation of radiophonic audio streams," in *Proc. of IEEE ICASSP*, Honolulu, Hawaii.

[20] J.-J. Aucouturier and F. Pachet, "Representing musical genre : A state of art," *Journal of New Music Research*, vol. 32, no. 1, 2003.

[21] ISMIR2004, "Audio description contest - genre/ artist id classification and artist similarity," 2004.

[22] MIREX, "Music information retrieval evaluation exchange," 2005, 2006, 2007.

[23] D. Jiang, L. Lu, H.-J. Zhang, J.-H. Tao, and L.-H. Cai, "Music type classification by spectral contrast," in *Proc. of ICME (IEEE Int. Conf. on Multimedia and Expo)*, 2002.

[24] J. Burred and A. Lerch, "A hierarchical approach to automatic musical genre classification," in *Proc. of DAFX*, London, UK, 2003.

[25] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Trans. on Speech and Audio Processing*, vol. 10, no. 5, pp. 293–302, 2002.

[26] C. McKay and I. Fujinaga, "Automatic genre classification using large high-level musical feature sets," in *Proc. of ISMIR*, Barcelona, Spain, 2004.

ANALYTICAL FEATURES FOR THE CLASSIFICATION OF PERCUSSIVE SOUNDS: THE CASE OF THE PANDEIRO

Pierre Roy, François Pachet, Sergio Krakowski

Sony CSL Paris
Paris, France
roy@csl.sony.fr
Pachet@csl.sony.fr
skrako@gmail.com

ABSTRACT

There is an increasing need for automatically classifying sounds for MIR and interactive music applications. In the context of supervised classification, we describe an approach that improves the performance of the general bag-of-frame scheme without loosing its generality. This method is based on the construction and exploitation of specific audio features, called analytical, as input to classifiers. These features are better, in a sense we define precisely than standard, general features, or even than ad hoc features designed by hand for specific problems. To construct these features, our method explores a very large space of functions, by composing basic operators in syntactically correct ways. These operators are taken from the Mathematical and Audio Processing domains. Our method allows us to build a large number of these features, evaluate and select them automatically for arbitrary audio classification problems.

We present here a specific study concerning the analysis of Pandeiro (Brazilian tambourine) sounds. Two problems are considered: the classification of entire sounds, for MIR applications, and the classification of attacks portions of the sound only, for interactive music applications. We evaluate precisely the gain obtained by analytical features on these two problems, in comparison with standard approaches.

1. ACOUSTIC FEATURES

Most audio classification approaches use either one of these two paradigms: a general scheme, called *bag-of-frames*, or ad hoc approaches.

The bag-of-frame approach ([2], also cited [41]) consists in considering the signal in a blind way, using a systematic and general scheme: the signal is sliced into consecutive, possibly overlapping frames (typically of 50ms), from which a vector of audio features is computed. The features are supposed to represent characteristic information of the signal for the problem at hand. These vectors are then aggregated (hence the "bag") and fed to the rest of the chain. First, a subset of available features is identified, using some feature selection algorithm. Then the feature set is used to train a classifier, from a database of labeled signals (training set). The classifier thus obtained is then usually tested against another database (test set) to assess its performance.

The use of the features as input to classifiers plays two roles: a dimension reduction role, and a representation role. Indeed, the signal itself could in principle be used as input to classifiers, but its dimension (number of samples) is usually too high with respect to the training set size, resulting in overfitting. Additionally, the time/amplitude representation of signals has long been acknowledged to be poorly adapted to represent perceptive information: audio features used in the classification literature aim precisely at capturing essential perceptive characteristics of audio signals that are not easily revealed in the temporal representation. A source of audio features is for instance MPEG7-audio ([15] or more specifically [28] or [20]) for the music domain. These features are usually of low dimensionality, and contain statistical information from the temporal domain (e.g. Zero-crossing rate), spectral domain (e.g. SpectralCentroid), or more perceptive aspects (such as sharpness, relative loudness, etc.).

The bag-of-frame approach has been used extensively in the MIR domain, for instance by [32]. A large proportion of MIR related papers has been devoted to studying the details of this chain of process: feature identification [28]; feature aggregation [34]; feature selection [26],[22],[7]; classifier comparison or tuning [1],[41].

An even larger proportion of ISMIR papers discuss the application of this approach to specific musical problems: genre classification [38],[21],[25],[39]; orchestral sound [27]; percussion instrument [37],[35],[13],[36]; tabla strokes [9],[6]; audio fingerprinting [5]; noises [12] as well as identification tasks, such as vocal identification [18] or mood detection [19].

This approach achieves a reasonable degree of success on some problems. For instance, speech music discrimination systems based on the bag-of-frame paradigm yield almost perfect results. However, the approach shows limitations when applied to more "difficult" problems. Although classification difficulty is hard to define precisely, it can be noted that problems involving classes with a smaller degree of abstraction are usually much more

difficult to solve. For instance, genre classification works well on abstract, large categories (Jazz vs. Rock), but performance degrades for more precise classes (e.g. Be-bop vs. Hard-bop).

In these cases, the natural tendency is usually to look for *ad hoc* approaches, which aim at extracting "manually" from the signal the characteristics most appropriate for the problem at hand, and exploit them accordingly. This can be done either by defining ad hoc features, integrated in the bag-of-frame approach (e.g. the 4-Hertz modulation energy used in some speech/music classifiers, [32], or by defining completely different schemes for classifying, e.g. the analysis-by-synthesis approach designed for drum sound classification [45], and further developed by [44] and [31].

One of the possible reasons for the limitation of bag-of-frame approach is that the generic features used, such as the Mpeg-7 feature set, do not always capture the relevant perceptive characteristics of the signals to be classified. Some classifier algorithms, such as kernel methods [33] including Support Vector Machines [4],[34] do try to transform the feature space with the aim of improving inter-class separability. However, the increasing sophistication of feature selection or classifier algorithms cannot compensate for any lack of information in the initial features set.

Although ad hoc approaches may indeed reach interesting performance, they are rarely reusable: ad hoc features are, by definition, problem specific. Consequently the scientific contribution (and epistemological status) of reports of ad hoc approaches is highly debatable.

In this work we try to extend the range of applications for which the general bag-of-frame approach gives satisfactory results, by proposing a mechanism that invents specific ad hoc features, in an automatic way to improve the classification performance.

To find better features than the generic ones, one can find inspiration in the way human experts actually invent ad hoc features. The papers quoted above use a number of tricks and techniques to this aim, combined with intuitions and musical knowledge. For instance, one can use some front-end system to normalize a signal, or pass it through some filter, add pre or post-processing to isolate the (hopefully) most salient characteristics of the signal.

We propose here to automate a process of feature invention, by an algorithm which explores quickly a very large space of ad hoc functions. The functions are built by composing together - in the sense of functional composition - elementary operators. We call these functions analytical because they are described by an explicit composition of functions, as opposed to other forms of signal reduction, such as arbitrary computer programs.

This paper is structured as follows: In Section 2 we introduce the EDS system, designed to create automatically and explore large sets of analytical features. Section 3 is devoted to the description of several experiments to compare the performance of analytical features against generic ones, on two sound classification problems for the Pandeiro (Brazilian percussion instrument): an easy one, for MIR applications, and a more difficult one, for interactive music applications.

2. CREATION OF ANALYTIC FEATURES: THE EDS SYSTEM

EDS – Extractor Discovery System – is developed at the Sony CSL laboratory in Paris [45] to study experimentally the notion of analytical feature for audio signal processing applications.

The EDS system is able to explore efficiently the space of analytical features for arbitrary supervised audio classification problem. A problem is determined by a database of audio samples labelled (usually by hand) with a finite set of classes. The exploration of the space of analytical features is based on various function creation methods from a set of basic operators, considered as elementary. These two aspects are described in the following sections.

2.1. A library of elementary operators

The choice of elementary operators is of course arbitrary. These operators were selected so as to allow the creation of functions with a "reasonable" degree of abstraction, i.e. represent salient perceptive characteristics of the sound with a small number of operators (about 10, see below), while allowing to create new, and possibly relevant functions. These operators are either basic mathematical operations (e.g. absolute value, max, mean) or signal processing operators such as Fourier transforms, filters, *Db*, and spectral operators like *spectralCentroid*, *spectralSkewness*. This library also includes more specifically musical operators such as *Pitch* or *Ltas* (Long Term Average Spectrum). For the sake of reproducibility, we describe in this paper results obtained with the 76 basic operators listed in Annex 1.

If we limit the size of analytical features we create (i.e. the number of operators used in its expression), we explore a finite function space. To give a rough idea of its size: the feature space of features composed of at most 5 operator contains $2{,}5.10^9$ functions. In practice, we explore functions of size at most 10, which represents a space of 5.10^{20} functions. Here are some typical examples of functions generated by EDS:

```
(A) Mean(Mfcc(Differentiation(x),5))
(B) Median(Rms(Split(Normalize(x),32)))
```

The first function (A) computes the average of the 5 first cepstral coefficients of the differentiation of the signal (represented by *x*). The second one (B) computes the mean value (*Median*) of the energy (*Rms*) of successive frames (*split*) of 32 samples long in the normalized signal.

Feature creation is controlled by two mechanisms:

1 – Each basic operator is typed according to the physical dimensions of its arguments. Types avoid creating syntactically meaningless features. For instance, the Fft operator takes as input something of the "time/amplitude" type, and its output type is "frequency/amplitude". EDS can therefore generate *Fft(HpFilter(x))*, but not, e.g., *Fft(max(x))*.

2 – Heuristics allow the system to further avoid creating unpromising functions. E.g. a heuristics penalizes functions with too many repetitions, like *Fft(Fft((Fft(x))))*.

In practice, adding a new basic operator to the library amounts to define 1) corresponding typing rules and 2) heuristics to control the use of this operator (see [24]).

2.2. Creating analytical features

The creation of analytical features by composing elementary operators is based on genetic programming search [16]. The main steps of this search are the following:

1. Construction of an initial population of analytical features, by random compositions of operators.
2. Evaluation: compute each feature on all the training signals, then use a classifier (see Section 2.3) to assess performance.
3. Iteration of the process. The next population is built from the best features found in the current population, to which are added new features obtained using various genetic transforms of the current features.

This genetic procedure explores parts of the infinite set of all analytical functions composed of basic operators. The convergence towards "meaningful" or "interesting" analytical features is not guaranteed as this heuristic-based approach can be entrapped into local minima.

The genetic transforms of step 3 are the following:

- *Substitution:* replacing one operators by another one with a compatible type. E.g.

```
(A') Max(Mfcc(Differentiation(x),5))
```

is a substitution (`Max` replaces `Mean`) of (A)

- *Cloning:* special case of substitution which consists in copying a feature but changing its parameters, e.g. :

```
(B') Median(Rms(Split(Normalize(x),64)))
```

is a clone of (B).

- *Mutation*: an extension of substitution to sub expressions appearing in the definition of a feature, which satisfies the typing rules:

```
(A") Mean(Chroma(Normalize(x)))
```

is a mutation of (A): sub expression `Chroma (Normalize(x))` replaces `Mfcc (Differentiation (x),5)`.

- *Crossover*: combining two features to create a new one while satisfying the typing rules. For instance:

```
(C) Mean(Rms(Split(Normalize(x),32)))
(C') Median(Rms(Split(Differentiation(x)))
```

are crossovers between (A) and (B).

- *Addition*: adding an operator to the root of a feature:

```
                (B")
Abs(Median(Rms(Split(Normalize(x),32))))
```

is an addition of (B).

2.3. Evaluation of features

To evaluate features, we need a computable criterion which measures the quality of a feature, i.e. its capacity to distinguish elements of different classes (labels). There are various ways to define such a criterion. The Fischer Discriminant Ratio [8] is often used because it is simple to compute and reliable for binary problems (two classes). However it is notoriously not adapted to multi-class problems, in particular for non convex distributions of data.

To improve feature evaluation, we chose to implement a "wrapper approach" to feature selection: features are evaluated using a classifier built during the feature search. The fitness is the performance of a classifier built with this unique feature (or more precisely its F-measure [30]) trained on the training database. This measure yields better performance than the Fischer criteria on multi-class problems.

3. PANDEIRO SOUND CLASSIFICATION

The Pandeiro is a Brazilian frame drum (a type of tambourine) used in particular in Brazilian popular music (samba, côco, capoeira, chôro). As it is the case for many popular music instruments, there is no official method for playing the Pandeiro. However, the third author, a professional Pandeiro player, has developed such a method, as well as a notation of the Pandeiro, that we use in this paper. This method is based on a classification of Pandeiro sounds in exactly six categories (see Figure 1):

Tung: Bass sound, also known as open sound;

Ting: Higher pitched bass sound, also open;

PA (big pa): A slap sound, close to the Conga slap;

pa (small pa): A medium sound produced by hitting the Pandeiro head in the center. Also considered as a slap, but softer;

Tchi: The jingle sound;

Tr: A tremolo of jingle sounds.

The need for automatically analyzing Pandeiro sounds is twofold. First, MIR applications, for education notably, require the ability to automatically transcribe Pandeiro solos.

Figure **1**. *The gestures to produce the six basic Pandeiro sounds.*

The second need is more original, and consists in developing real time interaction systems that expand the possibilities of the percussionist, to allow him to increase its musical "powers". In this case, we need to analyze robustly and quickly Pandeiro sounds, to trigger various events (see, e.g.[17]).

We therefore define two different analysis problems, corresponding to these two applications.

The first problem consists in classifying complete sounds (150ms duration) in the 6 classes. The second problem, much more difficult but more useful for real time applications, consists in classifying sounds using the least possible information, typically only the attack (about 3ms, that is 128 samples at 44 kHz), so as to allow a subsequent triggering of a musical event. To this aim we must build a reliable and very fast classifier.

3.1. Available sound databases

We have recorded a 2448 complete Pandeiro sounds (408 of each 6 types). They were produced with the same instrument and recorded on a Shure Beta 98 microphone linked to a MOTU Traveller sound card.

In order to classify the sounds, it is important to finely locate them in time. To this aim, we designed a robust attack identifier, which works as follows, on the sounds of the two databases.

We first extract an auditory spectrogram for the incoming signal [14]. Because of real-time constraints, we only compute an approximation of this spectrogram, as follows. The incoming signal is divided in non-overlapping frames of 1.4ms (64 samples at 44kHz). A loudness value is computed for each frame, generating the "loudness curve". We compute the differentiation of this curve. We call these two curves, the *loudness*" and the *differential*. Both are low pass filtered to reduce noise.

The attack detection is then performed in two phases. First we determine a threshold value for distinguishing actual sounds from noise. To this aim, the player captures 5 seconds of ambient noise (typically room noise as well as soft Pandeiro tchi sounds) and calculate the above mentioned curves from this audio information. The maximum value of these curves define the loudness and differential thresholds.

In the second phase, an attack is reported if, at a certain frame, the loudness level is greater than the loudness threshold and the norm of the differential curve exceeds the differential threshold. This frame is considered as the "attack frame".

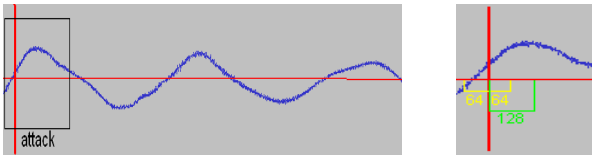


Figure **2**. *The attack detector: on the left, the full sound and attack portion. On the right, a zoom of the pre-attack and post-attack portions of the signal.*

When an attack is reported, two audio files are recorded. The first file is the audio contained both in the attack frame and its preceding frame. This file populates the pre-attack database (see **Figure 2**). We record another audio file with the audio stream right after the attack (the attack frame and one after it). This file populates the post-attack database.

Classifying the sound using only the pre-attack database information is the most difficult and useful problem in our context. The results on the post-attack database are slightly better, as it will be discussed, but they require an extra delay of 1.4ms (to get the next 64 samples) before processing.

3.2. Experiments: training and testing bases

In order to assess the efficiency of analytical features, we compare them to results obtained with a "reference feature set", whose complete list is given in Annex 2. This reference set includes general features commonly used in audio signal classification tasks, and well defined mathematically. The list includes notably the Mpeg-7 audio list, as well as several others, such as *Chroma*, often used for music analysis [10].
We systematically evaluate the performance of two classifiers: one built with the reference set, the other built with the features found by EDS with the set of basic operators in Annex 1.

Each experiment is in turn divided in two parts. First, classifiers are trained on training samples and tested on the test samples. To this aim, databases are systematically divided in two parts, 2/3 for the training, and 1/3 for the test. The samples are chosen randomly, to avoid artifacts (e.g. evolution of the membrane during the recording session, small variations in the player gestures).

In the second part, classifiers are trained and tested only on the test database, using 10-fold cross-validation.

This double experiment aims at showing that the advantages obtained by analytical features are consistent, and do not depend on the conditions of experiments. The cross-validation using only the test database is motivated by the fact the EDS already uses the training database for evaluating the analytical features. So reusing it for training the classifiers could produce biases (although we are not sure why and how).

Finally, for the attack problem, we build an experiment in which the signal itself is used as a feature (this is possible because these signals are very short). The aim is to confirm that the signal is not a good feature.

3.3. Choosing the classifiers

There is a vast literature on supervised learning algorithms [41] West, K., Cox, S., *Features and Classifiers for the automatic classification of musical audio signals*, ISMIR 2004.

[42] West, K., Cox, S., *Features and Classifiers for the automatic classification of musical audio signals*, ISMIR 2004.

[43] with no clear winner in general. To demonstrate the advantages of analytical features, we have conducted experiments with various classifiers, to avoid biases (e.g. SVM, *k*NN, J48). For the sake of clarity, we report here only the results with Support Vector Machines [34], which turned out to be the best and most stable algorithms tried. (We use the implementation provided in Weka [40] with the polynomial kernel.)

We used EDS in a fully automated way for the creation and selection of analytical features. For each problem, we ran the genetic search until no improvements were found in feature fit-

ness. For the complete sound problem, EDS evaluated about 40,000 features. For the attack problem EDS evaluated about 200,000 features.

3.4. Feature Selection

To compare the two approaches (general versus analytical features) in a fair manner, it is important to train classifiers on spaces with identical dimension. For the full sounds, all reference features (cf. Annex 2) could be computed, yielding a feature set of dimension 100. We have therefore selected 100 scalar analytical features among the 23,200 computed by EDS.

In the case of attacks, not all reference features were computable, because there is insufficient data: only 17 reference features could be computed and evaluated, with a total dimension of the feature set of 90. We therefore selected 90 analytical features among the 77,500 (resp. 53,500) EDS created for pre-attacks (resp. post-attacks).

To illustrate the results obtained, we have tried two different feature selection methods. Feature selection is important to avoid using redundant features. Here again, there are many feature selection methods [11] and the choice of the method turns out to be important for the final evaluation of the classifier. To avoid bias, we use, here also, two methods. The first is the IGR algorithm (Information Gain Ratio) [29]. Technically, this corresponds to the Weka *AttributeSelection* algorithm with the following parameters: the *evaluator* is a *InfoGainAttributeEval* and the *search* is a *Ranker*, which allows us to determine a priori the dimension of the feature set.

Secondly, we also developed a feature selection algorithm more suited to the application of EDS to multi-class problems. The idea is to select a feature set that "covers" optimally the classes to learn, from the viewpoint of individual features, that is, essentially of their F-measure (see Section 2.3). This algorithm iterates over all classes and selects successively features with the best F-measure for a given class.

Finally, we present results obtained for various sizes of feature sets (from 1 to 100). This is an important aspect in the context of real-time systems, where we want to minimize the number of features to compute in real time. As we will see, EDS finds not only better features but also feature sets of lesser dimension.

3.5. Results and comments

The tables Figure 3, Figure 4 and Figure 5 show the results obtained:

For the two problems, analytical features found by EDS improve the classification performance. The full sound problem is relatively easy. The use of the full reference feature set (dimension 100) yields a precision of about 99,9%. With the same dimension, analytical features yields the same precision. The gain becomes interesting if we consider feature sets of smaller dimension: 2 analytical features yield a precision of 89,5% versus 78% for general features.

Experiment Description			Feature Set Dimension										
			100	90	75	50	25	15	10	5	3	2	1
Reference	*IGR*	*Train/Test*	99,9	99,9	99,6	99,5	99	99,5	99,1	92,8	88,5	65,2	56
Reference	*IGR*	*10-fold XV*	99,9	99,5	99,5	99,5	99,1	98,6	98,4	92	82	60,5	59,3
EDS	*IGR*	*Train/Test*	99,9	99,9	98,5	98,3	98,9	98,3	99,1	98	68,9	36,1	36,9
EDS	*IGR*	*10-fold XV*	99,9	99,9	99,9	98,8	98	98,4	98,2	97,8	64,7	36	21,2
Reference	*EDS FS*	*Train/Test*	99,9	99,9	99,9	99,8	99,1	99,1	98,9	98,8	93,6	80,8	67,2
Reference	*EDS FS*	*10-fold XV*	99,9	99,6	99,6	99,4	98,6	98,4	98,8	98,3	93,4	78,1	61,6
EDS	*EDS FS*	*Train/Test*	99,9	99,9	98,9	99,9	99,9	99,6	99,5	99	89,9	88,8	73,8
EDS	*EDS FS*	*10-fold XV*	99,9	99,9	98,9	99,7	99,6	99,5	99,4	99	91,3	89,5	73,6

Figure **3**. *Results on full sounds.* ***IGR*** *stands for Information Gain Ratio.* ***EDS FS*** *denotes our feature selection algorithm based on the F-measure.* ***Train/Test*** *denotes the experiment in which the classifier is trained on the training database and tested on the test database.* ***10-fold XV*** *denotes the 10-fold cross validation experiment on the test database.*

Experiment Description			Feature Set Dimension									
			90	75	50	25	15	10	5	3	2	1
Reference	*IGR*	*Train/Test*	94,8	95,6	93	76,8	76	76,1	73,5	65,9	54,2	44,6
Reference	*IGR*	*10-fold XV*	94,8	94,8	92,7	78,8	73,2	72,2	66,2	65,2	48,3	43,3
EDS	*IGR*	*Train/Test*	94,8	95,6	92,9	81	76,6	76	69,6	65,7	54,2	44,5
EDS	*IGR*	*10-fold XV*	95,1	94,5	92,8	78,8	73,2	73,5	66,8	65,3	50,9	45
Reference	*EDS FS*	*Train/Test*	94,7	94,7	94,8	92,4	90,8	88,7	87,2	84,1	71,4	52,4
Reference	*EDS FS*	*10-fold XV*	95,4	94,7	94	91,9	90,8	87,9	85,7	81,5	68,5	51,3
EDS	*EDS FS*	*Train/Test*	96	95,5	95,1	93,9	93,5	93,4	93	89,9	86,2	71,7
EDS	*EDS FS*	*10-fold XV*	95,1	95	95,2	93,3	92,9	92,5	92,5	88,3	84,8	71,6
Signal			75.8	75.8	72.5	67.6	67.1	46	44	36.6	37	35.5

Figure **4**. *Results obtained with on pre-attacks. See above for abbreviations. The "Signal" line gives the performance of classifiers using the input signal directly as a feature.*

Experiment Description			Feature Set Dimension									
			90	75	50	25	15	10	5	3	2	1
Reference	*IGR*	*Train/Test*	91,8	91,3	89,6	76,6	78,3	67,5	64,3	56,1	51,1	49
Reference	*IGR*	*10-fold XV*	92,6	91,2	88,8	79,9	73,2	67,4	64,7	44,2	42,4	34,5
EDS	*IGR*	*Train/Test*	95,1	93,3	92,3	77,7	72,5	63	61,3	54,7	54,5	56,9
EDS	*IGR*	*10-fold XV*	94,9	93,8	92,4	80,8	78,9	62,4	61	55,1	55,9	54,9
Reference	*EDS FS*	*Train/Test*	91,9	91,5	91	87,7	86,7	83,4	83,6	71,7	55,6	43,9
Reference	*EDS FS*	*10-fold XV*	91,9	91,5	90,2	86,1	85,2	78,9	82	68,5	48,6	39
EDS	*EDS FS*	*Train/Test*	94,9	94,4	94	92,1	91,4	87,9	90,1	88,6	80,4	72,1
EDS	*EDS FS*	*10-fold XV*	94,5	94	93,3	91,4	91,4	89	89,5	88	80,1	69,2
Signal			77.7	76.9	73.3	64.1	64.2	60	59.2	58.1	57.5	44

Figure **5**. *Results obtained with on post-attacks. See above for abbreviations.*

The attack problems are more difficult and interesting. Analytical features are still better than general ones, in particular for small feature sets. For the post-attack problem, 3 analytical features perform as well as the 50 best general features.

We can note that the gain evolution depends on the feature selection algorithm used. The standard IGR algorithm does not select the best EDS features for small size feature sets (this result is already known, see [3]). However, our feature selection algorithm yields better results for all sizes of the feature set, as illustrated in Figure 6. This result shows again, if needed, the difficulty in interpreting the precision of classifiers directly.

The performance gain brought by analytical features for small feature sets has a lot of advantages, in particular for real-time applications. For the attack problem, 3 features yield a precision greater than that obtained with 50 reference features. These features are the following:

Abs (*Log* (*Percentile* (*Square* (*BpFilter* (x, 764, 3087)), 64)))

Centroid (*MelBands* (*Differentiation* (*HpFilter* (*Power* (*Normalize* (x), 3), 100)), 6))

Abs (*Sum* (*Arcsin* (*Mfcc* (*Hann* (*HpFilter* (x, 19845)), 20))))

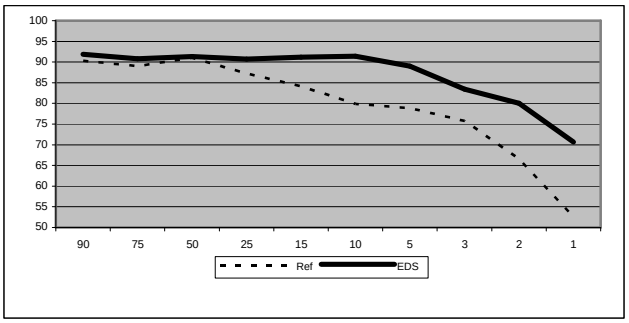


Figure **6**. *Analytical vs. reference features on attacks*

This particular result allows us to consider real-time implementations: on a 3GHz Pentium IV PC, the computation of the 3 features for a 2,8 ms signal takes about 3 ms, to be compared to the computation of 50 generic features, which takes 12 ms, that is 4 times slower.

4. CONCLUSION

We have presented a method for creating audio features, called analytical, by composing basic signal operators, to improve the performance of classification algorithms. We have illustrated this idea on audio classification problems dealing with Pandeiro sounds. In all cases (classifying full sounds, or only portions of the attacks) analytical features do improve the performance of classification, as compared to results obtained with generic, Mpeg-7 like features, in a bag-of-frame approach. The gain is notable both in terms of classification precision and feature set size. Moreover, analytical features improve classifi-

cation algorithms independently of any other optimization process (such as boosting, bagging or *ad hoc* approaches).

5. ACKNOWLEDGMENTS

The work of Sergio Krakowski is partially supported by a CAPES scholarship.

6. REFERENCES

[1] Aucouturier, J.-J. and Pachet, F. Tools and Arc*hitecture for the Evaluation of Similarity Measures : Case Study of Timbre Similarity*. ISMIR 2004.

[2] Aucouturier, J.-J, Defreville, B. and Pachet, F. The bag-of-frame approach to audio pattern recognition: A sufficient model for urban soundscapes but not for polyphonic music. Journal of the Acoustical Society of America, 2007.

[3] Blum , A. and Langley, P. *Selection of Relevant Features and Examples in Machine Learning*, Artificial Intelligence, 1997 pp.245-271, Dec. 1997.

[4] Boser, B. Guyon, I. and Vapnik V. *A training algorithm for optimal margin classifiers*. In D. Haussler, editor, 5th Annual ACM Workshop on COLT, pp.144-152, Pittsburgh, PA. ACM Press. 1992.

[5] Cano, P., Batlle, E., Kalker, T., and Haitsma, J. *A Review of Audio Fingerprinting*. J. VLSI Signal Process. Syst. 41, 3 (Nov. 2005), 271-284. 2005.

[6] Chordia, P. Segmentation and Recognition of Tabla Strokes, ISMIR, pp. 107-114, 2005.

[7] Fiebrink, R. and Fujinaga, I. *Feature Selection Pitfalls and Music Classification*. ISMIR 2006, pp. 340-341, 2006.

[8] Fisher, R. A. *The use of Multiple Measurements in Taxonometric Problems* Ann. Eugenics, vol. 7, pp. 179-186. 1936.

[9] Gillet, O. and Richard, G. *Automatic Labelling of Tabla Signals*. ISMIR 2003.

[10] Gomez Gutierrez E. *Tonal Description of Music Audio Signals*, PhD Thesis, Universitat Pompeu Fabra, Barcelone, 2006.

[11] Guyon, A. Elisseeff, *An Introduction to Variable and Feature Selection*, Journal of Machine-Learning Research, 3 1157-1182, 2003.

[12] Hanna, P., Louis, N., Dessainte-Catherine, M. and Benois-Pineau, J. *Audio Features for Noisy Sound Segmentation*. ISMIR 2004, 2004.

[13] Herrera, P., Yeterian, A. and Gouyon, F. *Automatic classification of drum sounds: a comparison of feature selection methods and classification techniques*. Proceedings of 2nd International Conference on Music and Artificial Intelligence, Edinburgh, Scotland. 2002.

[14] Jehan, T. *Hierarchical Multi-Class Self Similarities* IEEE WASPAA, 2005.

[15] Kim, H.G., Moreau, N. and T. Sikora Mpeg7 *Audio and Beyond: Audio Content Indexing and Retrieval*. Wiley & Sons. 2005.

[16] Koza, J. R. Genetic Programming: on the programming of computers by means of natural selection, Cambridge, MA: The MIT Press. 1992.

[17] Krarkowski, S. Pandeiro+, music video available at: http://www.skrako.com/eng/pop_video.html?aguas

[18] Lay Nwe, T. and Wang, Y. *Automatic Detection Of Vocal SegmentsiIn Popular Songs*. ISMIR 2004.

[19] Liu, D., Lu, L. and Zhang, H.J., *Automatic mood detection from acoustic music data*, ISMIR 2003.

[20] McEnnis, D. McKay, C., Fujinaga, I. Depalle, P. *jAudio: a feature extraction library*, Ismir 2005.

[21] McKay, C. and Fujinaga, I. Automatic Genre Classification Using Large High-Level Musical Feature Sets. ISMIR 2004.

[22] McKinney, M.F. and Breebart, J. *Features for audio and music classification.* ISMIR 2003.

[23] Meng, A. and Shawe-Taylor, J. *An Investigation of Feature Models for Music Genre Classification Using the Support Vector Classifier*. ISMIR 2005.

[24] Pachet, F. and Zils, A. *Automatic Extraction of Music Descriptors from Acoustic Signals*. ISMIR 2004.

[25] Elias Pampalk, Arthur Flexer & Gerhard Widmer Improvements of Audio-Based Music Similarity and Genre Classification (pp. 628-633), ISMIR 2005

[26] Peeters, G. and Rodet, X. *Automatically selecting signal descriptors for sound classification*. Proceedings of the 2002 ICMC, Goteborg (Sweden). 2002.

[27] Peeters, G. Automatic Classification of Large Musical Instrument Databases Using Hierachical Classifiers with Inertia Ratio Maximization, 115th AES Convention New-York, NY, USA, 2003.

[28] Peeters, G. *A large set of audio features for sound description in the Cuidado project*. http://recherche.ircam.fr/equipes/analyse-synthese/peeters/ARTICLES/Peeters_2003_cuidadoaudiofeatures.pdf

[29] Quinlan, J.R. *C4.5: Programs for machine learning*. Morgan Kaufmann. 1993.

[30] Rijsbergen, C. J. van *Information Retrieval*. Butterworths, London. 1979

[31] Sandvold, V. Gouyon, F. Herrera, P. Percussion classification in polyphonic audio recordings using localized sound models, ISMIR 2004.

[32] Scheirer, Eric D. and Slaney, Malcolm *Construction and evaluation of a robust multifeature speech/music discriminator*. Proc. ICASSP '97. 1997.

[33] Schölkopf, B. and Smola, A. *Learning with Kernels*, MIT Press, Cambridge, MA. 2002.

[34] Shawe-Taylor, J. and Cristianini, N. Support Vector Machines and other kernel-based learning methods - Cambridge University Press. 2000.

[35] Sinyor, E., McKay, C., Fiebring, R., McEnnis, D. and Fujinaga, I. *Beatbox Classification Using ACE*. ISMIR 2005: 672-675, 2005.

[36] Steelant, D. van Tanghe, K. Degroeve, S. De Baets, B. Leman, M. and Martens, J.-P. *Classification of percussive sounds using Support Vector Machines*, Proceedings of the annual machine learning conference of Belgium and The Netherlands, 2004.

[37] Tindale, A.R., Kapur, A., Tzanetakis, G. and Fujinaga, I. *Retrieval of percussion gestures using timbre classification techniques*, ISMIR 2004,2004.

[38] Tzanetakis, G. *Automatic Musical Genre Classification of Audio Signals*, ISMIR 2001.

[39] Tzanetakis, G. Cook, P. *Musical Genre Classification*, IEEE Transactions on Speech and Audio Processing, Vol. 10, No. 5, July, pp. 293-301. 2002.

[40] Weka, Data Mining Software in Java, see http://www.cs.waikato.ac.nz/ml/weka/

[41] West, K., Cox, S., *Features and Classifiers for the automatic classification of musical audio signals*, ISMIR 2004.

[42] West, K., Cox, S., *Features and Classifiers for the automatic classification of musical audio signals*, ISMIR 2004.

[43] Witten, I.H. Eibe, F. *Data Mining: Practical Machine Learning Tools and Techniques.* M. Kaufmann Publisher, 2nd Edition. 2005.

[44] Yoshii, K., Goto, M., and Okuno, H.G. AdaMast: *A Drum Sound Recognizer based on Adaptation and Matching of Spectrogram Templates*, ISMIR 2004.

[45] Zils A., Pachet F., Delerue O., Gouyon F. *Automatic Extraction of Drum Tracks from Polyphonic Music Signals*. Proceedings of WEDELMUSIC, Dec. 2002.

[46] Zils, A. *Extraction de descripteurs musicaux: une approche évolutionniste*, PhD, Univ. Paris 6. 2004.

7. ANNEXES

All the sounds and results of this study are made available to interested readers, as well as feature files (Weka format): http://SecondAuthorWebSite/pandeiro

7.1. Annex 1 – Basic EDS operators

The list of basic operators used by EDS in this study is the following:

Abs	HarmSpectralVar	PeakPos
Arcsin	HFC	Percentile
AttackTime	HMean	Pitch
Autocorrelation	HMedian	PitchBands
Bandwidth	HMax	Power
BarkBands	HMin	Range
Bartlett	HpFilter	RHF
Blackman	Integration	Rms
BpFilter	Inverse	SpectralCentroid
Centroid	Iqr	SpectralDecrease
Chroma	Length	SpectralFlatness
Correlation	Log10	SpectralKurtosis
dB	LpFilter	SpectralRolloff
Differentiation	Max	SpectralSkewness
Division	MaxPos	SpectralSpread
Envelope	Mean	Split
Fft	Median	SplitOverlap
FilterBank	MelBands	Sqrt
Flatness	Min	Square
Hamming	Mfcc0	Sum
Hann	Mfcc	Triangle
Hanning	Multiplication	Variance
HarmSpectralCentroid	Normalize	Zcr
HarmSpectralDev	Nth	Harmonicity(Praat)
HarmSpectralSpread	NthColumns	Ltas(Praat)

A precise description of operators can be found in [46].

7.2. Annex 2 – Reference features

The list of general features used as the reference set is the following (features preceded by '*' could not be computed on the attack sounds because of their size):

```
* HarmonicSpectralCentroid(Hanning(x))
* HarmonicSpectralDeviation(Hanning(x))
* HarmonicSpectralSpread(Hanning(x))
Log10(AttackTime(x))
*Pitch(Hanning(x))
SpectralCentroid(Hanning(x))
* SpectralFlatness(Hanning(x))
SpectralSpread(Hanning(x))
Centroid(x)
PitchBands(Hanning(x),12.0)
Mfcc0(Hanning(x),20.0)
* HarmonicSpectralVariation(SplitOverlap(Hanning(x),2048,0.5))
Rms(x)
RHF(Hanning(x))
HFC(Hanning(x))
SpectralKurtosis(Hanning(x))
SpectralSkewness(Hanning(x))
SpectralRolloff(Hanning(x))
Iqr(x)
Chroma(Hanning(x))
MelBands(Hanning(x),10.0)
BarkBands(Hanning(x),24.0)
Zcr(x)
```

AUTOMATIC MUSIC DETECTION IN TELEVISION PRODUCTIONS

Klaus Seyerlehner, Tim Pohle, Markus Schedl
Dept. of Computational Perception
Johannes Kepler University Linz, Austria
klaus.seyerlehner@jku.at

Gerhard Widmer
Dept. of Computational Perception
Johannes Kepler University Linz, Austria and
Austrian Research Institute for AI, Vienna
gerhard.widmer@jku.at

ABSTRACT

This paper presents methods for the automatic detection of music within audio streams, in the fore- or background. The problem occurs in the context of a real-world application, namely, the analysis of TV productions w.r.t. the use of music. In contrast to plain speech/music discrimination, the problem of detecting music in TV productions is extremely difficult, since music is often used to accentuate scenes while concurrently speech and any kind of noise signals might be present. We present results of extensive experiments with a set of standard machine learning algorithms and standard features, investigate the difference between frame-level and clip-level features, and demonstrate the importance of the application of smoothing functions as a post-processing step. Finally, we propose a new feature, called *Continuous Frequency Activation* (CFA), especially designed for music detection, and show experimentally that this feature is more precise than the other approaches in identifying segments with music in audio streams.

1. INTRODUCTION

Annotation and tagging of audio data have mainly been human tasks in the past. The growing amount of digital media, however, makes manual tagging impractical. One of these tiresome tasks is to determine whether or not music is present in an audio excerpt. This problem occurs in many application contexts. A particular variant of this problem was posed to us by the Austrian National Broadcasting Corporation (ORF): the task is to automatically determine where in the sound track of a TV production there is music being played, in the foreground or in the background. This is important for the calculation of royalty fees, which are paid to a national agency according to certain rules. Ideally, the production team would supply a precise list of all the music segments occurring in a TV production, but in reality these lists are often incorrect or simply empty, which requires the ORF to more or less guess the amount of music within a production, since manually annotating all productions is simply impossible. Thus, it would be desirable to have a system that automatically detects music segments and predicts, with high precision, the percentage of time where music is present within a production.

In this paper, we present our approach to this difficult music detection problem. First a brief literature review is given out in Section 2. Section 3 presents an overview of our overall approach and a detailed description of the features examined. In Section 4.1 we report on the ground truth data that were collected, on extensive machine learning experiments and the results obtained with them. We then introduce a new feature in Section 5 and show that this feature indeed yields further improvement. Finally, we present our conclusions and discuss future work.

2. RELATED WORK

There has been quite some research recently on the automatic discrimination between speech and music. Even if our problem is related, it must be pointed out that detecting music within TV productions is more complicated than simple music/speech discrimination. The major reason is that music and other sounds are generally mixed in TV, and in particular that the musical background of movies is typically rather soft compared to spoken words or scene-related sounds in the foreground. That is because music is normally used to create the atmosphere of a scene and should not attract the listener's attention. Interestingly, when people pay attention to the presence of music in movies, most of them are surprised at what a high percentage of music is present and how difficult it is, in many cases, to even tell whether or not music is being played at all.[1] Thus, in contrast to the typical datasets usually used in speech/music discrimination research, which mostly consist of relatively distinct cases of the classes *music* and *speech*, we mainly have to deal with soft music signals mixed with other sound signals.

There has been some previous work that is relevant to our problem, for example Santo et al. [1], who worked on automatic video segmentation based on audio track analysis. In contrast to our problem – deciding whether there is music present or not – they deal with seven different classes. When aggregating the results they report for their 7 classes to the two broad classes *music* and *no_music* only, we arrive at a classification accuracy of their system of approximately 75.86%, which we consider as a quite good and a useful baseline to evaluate our approach. Khan et al. [2] give an interesting overview of existing features and methods for movie audio classification, although the results of the various approaches are incomparable to each other, due to the lack of common test databases and different application areas. Minami et al. [3, 4] focus on the automatic indexing of videos by discriminating video scenes according to the classes *speech*, *music* and *music and speech*. Their system is composed of two expert systems, one for detecting music and the other one for detecting speech. They report an average detection rate of 90% for musical segments. However their ground truth database seems to be very unrepresentative – we re-implemented their approach and only achieved a low 55.78% on our real world dataset (see below). Mauclair and Pinquier [5] apply their speech/music classification system to recordings from radio stations, where they achieve a classification accuracy of 86.9% for music/non-music discrimination (which should

[1] To illustrate the difficulty of this problem we provide, on our homepage, some audio samples of the television productions we have been annotating — see http://www.cp.jku.at/people/seyerlehner/md.html.

be simpler than background music detection in TV shows). Altogether, speech/music discrimination seems to have broad application potential and attracted a lot of research attention, but to our knowledge there is no scientific work focusing specifically on music detection.

3. SYSTEM OVERVIEW

The architecture of our music detection system resembles a classical machine learning process extended by a post-processing stage. In a first step the audio stream is cut into small frames and features are extracted for each frame. Second, a classifier is trained on a distinct training set and learns to distinguish the two classes *music* and *no_music*. It is then used to predict the class labels for all the frames in new TV shows. In a final post-processing stage the classification results are smoothed in such a way that we obtain a plausible label sequence for longer continuous segments of audio.

Since the choice of features is very critical, we first decided to test some promising features known from recent work in the field of speech/music discrimination. The next section gives a detailed description of the features we have chosen to investigate.

3.1. Features

We focused on four sets of features. A major aspect during the decision process was that a feature must still be able to capture musical properties, even if speech or any kind of sounds are present. Thus, for example, we did not consider *4 Hz modulation energy* and *zero-crossing rate* related features, since they are merely useful in speech/music discrimination to detect speech segments, but not in the case of music detection alone.

3.1.1. Spectral Entropy (SE)

In general the entropy measures the uncertainty or unpredictability of a probability mass function (PMF). The entropy of the spectrum of an audio frame is a well-known feature for speech/music discrimination [6]. To be able to compute the entropy, the power spectrum is converted into a probability mass function:

$$x_i = \frac{X_i}{\sum_{j=1}^{N} X_j} \tag{1}$$

where X_j denotes the energy of j-th frequency component of the STFT spectrum of the current frame. For each frame the entropy is computed from $\vec{x}$ as:

$$H = \sum_{i=1}^{N} -x_i \log_2 x_i \tag{2}$$

In general the *spectral entropy* should be higher for speech frames than for music frames.

3.1.2. Chromatic Spectral Entropy (CSE)

The Chromatic Spectral Entropy, as defined in [7], is a variant of the *Spectral Entropy*. Instead of computing the entropy directly based on the normalized power spectrum, the power spectrum is first mapped onto the Mel-frequency scale and divided into 12 sub-bands, where the center frequency f_i of a band coincides with one of the 12 semitones of the chromatic scale. For a fixed center frequency f_0 of the lowest band, the center frequencies of the other sub-bands correspond to:

$$f_i = 1127.01048 * \log(\frac{f_0 * 2^{\frac{k}{12}}}{700} + 1) \tag{3}$$

As for the *Spectral Entropy* the energies of the sub-bands X_i are normalized according to equation (1), and the entropy of the chromatic representation of a frame is again computed as in equation (2).

3.1.3. Mel Frequency Cepstrum Coefficients (MFCC)

Mel Frequency Cepstrum Coefficients are a compact representation of the spectral envelope of a frame. After a non-linear mapping onto the Mel-frequency scale, to better approximate the frequency resolution of the human ear, the envelope of the log-spectrum is compactly represented by the first few coefficients after a DCT compression. MFCCs are well-known for capturing timbral aspects of short audio frames. Ezzaidi et al. [8] show the successful application of ΔMFCCs in the area of speech/music classification.

3.1.4. Linear predictive Coefficients (LPC)

Linear prediction is used to predict the current value $\hat{s}(n)$ of the real-valued time series $s(n)$ based on past p samples [9].

$$\hat{s}(n) = \sum_{i=1}^{p} a_i s(n-i) \tag{4}$$

The filter coefficients a_i define the p-th order linear predictor (FIR filter). The optimal filter coefficients are determined by minimizing the prediction error in the least squares sense. The prediction error, or residual error, is given by

$$e(n) = s(n) - \hat{s}(n) = s(n) - \sum_{i=1}^{p} a_i s(n-i). \tag{5}$$

For the compact representation of an audio frame the time-series $s(n)$ is the time-domain sample sequence of the current frame. The prediction error is expected to be significantly higher for impulsive speech compared to steady notes played by instruments.

Additionally, ΔMFCC, ΔLPC, ΔSE and Δ CSE were also added to the feature set.

3.2. From frame-level to "clip-level" features

Short-term frame-level features capture essential information about the sound of an audio frame. Such an audio frame commonly lasts 10-40 ms, which means that it contains little if any temporal information. Our machine learning approach does not assume any specific ordering of the training or test examples either and thus most of the temporal information is lost. For speech/music discrimination temporal information might be useful, because speech segments tend to be more impulsive than music segments, leading to a higher variance of the frame-level feature values over time. To capture some of this temporal information we summarize a sequence of consecutive feature vectors by computing mean and standard deviation over a fixed number of frames. The resulting features – now representing an audio clip of several seconds of audio – are called *clip-level* features in accordance with [9]. We performed dedicated machine learning experiments to investigate if these clip-level features yield any improvement over frame-level features. We will report on the results in section 4.1.

3.3. Smoothing

The result of the classification process is a sequence of class labels *music* or *no_music*, where each label is associated with a short excerpt of audio. Depending on the type of feature, either frame-level or clip-level, the labels might change every few milliseconds or every few seconds. An analysis of our annotated ground truth material (see section 4.1) shows that there are no music segments shorter than 3 seconds, and only 14 out of 324 music segments are shorter than 7 seconds. This indicates that music as used in TV productions lasts at least several seconds. Consequently, frame-based class labels should be aggregated into larger continuous segments of *music* or *no_music* in order to get a plausible segmentation of an audio stream. To come up with a smoothed version of the label sequence we iteratively apply (twice) a majority filter with a sliding window length corresponding to 5 seconds. In a final step, if there are any continuous label segments left that are shorter than 5 seconds, we remove them by swapping first of all the *no_music* segments shorter than 5 seconds to *music* and then the *music* segments shorter than 5 seconds to *no_music*. Altogether we smooth the label sequence in a fist step and filter out all remaining segments shorter than 5 seconds by swapping their class label. It is important to note that smoothing functions might introduce a bias by slightly favoring one of the classes.

4. EXPERIMENTS AND RESULTS

4.1. Data and Ground Truth

To be able to train our classifiers on real world situations representative for the later operation at the television station, we recorded a number of real TV telecasts. Recording was done using the DVB-T standard, and the digital digital video streams were encoded as MPEG-2. In a second stage the sound tracks of the 13 TV shows (approximately 545 minutes of audio) were converted to PCM mono at 22 kHz with a precision of 16 Bit/sample. Thereafter all audio files were annotated manually according to the class labels *music* and *no_music*, which turned out to be quite challenging, because it is often hard to tell when precisely some background music starts or stops playing. Consequently, assuming an imprecision for each change of the label of just one second, we get an upper bound on the overall classification accuracy of 98%, which is still a very optimistic estimate. Table 1 shows the distributions of the two classes for each of the 13 recorded ORF TV productions. Obviously, the amount of music present in a show depends heavily on the type of show. The baseline for the overall classification accuracy is 58.01%, which is the percentage of the more frequent class, *no_music*.

To yield a clear separation between training and test data, all the frames of an entire show must either be in the training or the test set. The first three shows, which are separated from the others in table 1, constituted the training set. Such a separation prevents a bias of learning algorithms towards specific characteristics of a single broadcast, e.g. the voice of the moderator, which would lead to too optimistic results.

title	type	% music	min
Der Volksanwalt	law show	1.48 %	35
Starmania	music show	50.18 %	89
Sturm der Liebe	soap opera	70.52 %	49
Alpen Donau Adria	documentary	57.08 %	30
Barbara Karlich Show	talk show	7.51 %	57
Da wo es noch Treue gibt	soap opera	62.90 %	89
Frisch gekocht	cooking show	10.01 %	24
Gut beraten Österreich	talk show	8.76 %	18
Heilige Orte	documentary	54.34 %	44
Heimat fremde Heimat	documentary	29.72 %	30
Hohes Haus	parliament show	17.50 %	30
Julia	soap opera	80.36 %	43
ZIB	news show	4.91 %	7
total	–	41.99 %	545

Table 1: *The ground truth data.*

4.2. Prediction Experiments

One of our interests was to find out if we can achieve any improvement by using clip-level features generated out of frame level features instead of using the frame-level features themselves (see section 3.2). To do so, we extracted both frame-level (with a window size of 24ms) and clip-level features (with a window size of 1172ms) for all 13 audio streams. Our current framework makes use of the `WEKA`[10] machine learning library. We used five (very) different `WEKA` classifiers to evaluate the features via machine learning experiments. The simple nearest-neighbor classifier *IBk* was chosen as a representative of instance-based learning methods, *Support Vector Machines (SMO)* for kernel-based machine learning methods, *MultilayerPerceptron* as the most popular representative of the neural network family of classifiers, and *REPTree* and *RandomForest* for decision tree learners. For each of these classifiers we computed the overall classification accuracy on the test set (approximately 372 minutes of audio) after learning from the independent training set.

classifiers	frame level	clip level
IBk	**69.94** %	66.47 %
MultilayerPerceptron	**69.67** %	65.99 %
SMO	69.48 %	**73.27** %
REPTree	64.07 %	**64.48** %
RandomForest	70.66 %	**73.19** %

Table 2: *Frame level versus clip level features.*

Table 2 shows the results. They seem to strongly depend on the type of classifier. No general advantage of clip-level features compared to frame-level features could be shown by our experiments. All further experiments are based on frame-level features.

The second experiment investigated the benefits of smoothing. The classification results before and after the application of the smoothing function are compared in table 3. For all of the five classifiers a substantial improvement of the classification result could be shown.

In general, applying smoothing functions increases the accuracy, but tests with various smoothing functions indicate that more sophisticated smoothing does not seem to improve the classification results any further. Figure 4 shows the classification results of *"Julia"* before and after the application of the smoothing function. Even visually it is quite obvious that the aggregation of the frame level classifications makes sense.

classifier	no smoothing	smoothed
IBk	69.94 %	**79.82** %
MultilayerPerceptron	69.67 %	**81.21** %
SMO	69.48 %	**76.19** %
REPTree	64.07 %	**73.48** %
RandomForest	70.66 %	**77.20** %

Table 3: *Smoothed versus original results.*

The best overall result using the machine learning approach and various standard features was achieved with the smoothing function applied to the class predictions of the *MultilayerPerceptron*. A total accuracy of **81.21 %** can be reached with this configuration. Table 4 shows the classification accuracy for each recorded show of the test set separately.

title	% real	% est.	diff.
Alpen Donau Adria	57.08 %	19.00 %	**38.08**
Barbara Karlich Show	7.51 %	12.33 %	4.82
Da wo es noch Treue gibt	62.90 %	63.47 %	0.57
Frisch gekocht	10.01 %	22.73 %	**12.72**
Gut beraten Österreich	8.76 %	6.42 %	2.34
Heilige Orte	54.34 %	49.82 %	4.52
Heimat fremde Heimat	29.72 %	52.17 %	**22.45**
Hohes Haus	17.50 %	15.84 %	1.66
Julia	80.36 %	68.01 %	**12.35**
ZIB	4.91 %	2.84 %	2.07

Table 4: *The percentage of music really present versus the percentage estimated.*

Since in our project we have to determine the *percentage* of time where music is present within a production, the difference in percentage points is our real quality measure. Even if the machine learning approach yields an overall classification accuracy of more than 80%, the error, in terms of the difference in percentage points, is too high for some TV shows to be useful for the ORF. In general, a maximal prediction error of 5 percentage points would be desirable for the planned application, and a prediction error of 10 percentage points may be the maximum that is still considered acceptable. In table 4 all results exceeding this maximum error of 10 percentage points are highlighted. Consequently, to further improve the obtained results, a new feature especially designed for the detection of music was developed and will be introduced in the next section.

5. CONTINUOUS FREQUENCY ACTIVATION (CFA) - A NEW FEATURE FOR MUSIC DETECTION

Our experiments show that standard speech/music discrimination features work reasonably well overall, but produce rather large errors in some cases. On the other hand most of these features were not designed for this particular type of music detection task we are working on. None of these features accounts for the special characteristics of music signals. In essence, what makes music different from other sounds are structural properties. Examples of higher-level structural properties are rhythm and harmony. Music

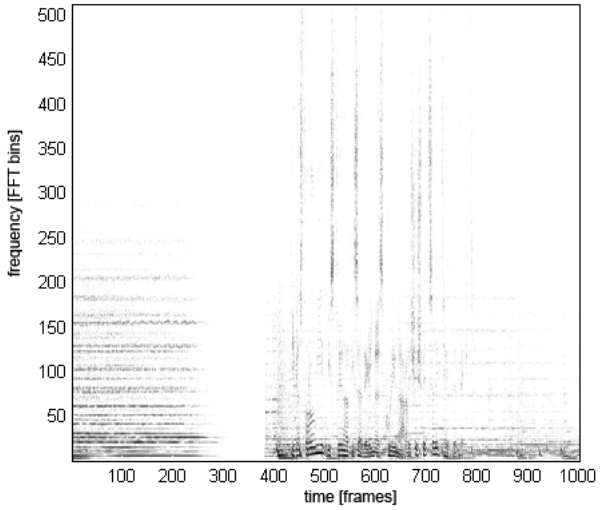


Figure 1: *Spectogram of an audio excerpt containing music and speech.*

detection might benefit from focusing on such structural properties, at various levels of the signal.

Consider for example the audio track of a movie containing some sort of background music. Because of the music being played in the background the music signal itself will be embedded very softly in the audio signal, and the global characteristics of the audio signal will more strongly resemble the characteristics of speech or noise. Features characterizing for example the frequency distribution of an audio frame will tend to model the properties of the sounds belonging to the foreground and are therefore not useful for music detection in such a case. However, we still might be able to reveal structural properties of background music, e.g. the rhythmic structure, because it is unlikly that all rhythmic events are completely masked by the foreground signal. Consequently, features focusing on the extraction of structural properties especially attributable to music might be more successful in separating *music* from *no_music* segments. In the following, we develop an intuitive feature that is meant to capture a kind of low-level structural property of musical sounds.

5.1. The basic Idea

In general music tends to have more stationary parts than speech, resulting in horizontal perceivable bars within the spectrogram representation of an audio signal (see figure 1). This property was already investigated by Hawley, who was interested in the structure of music [11] and who was the first to propose a simple *music detector* based on this. The horizontal bars in the spectrogram are continuous activations of specific frequencies and are usually the consequence of sustained musical tones. Minami et al.[3, 4] tried to construct an improved feature based on this observation. Their feature seems to work quite well for clearly distinct examples of *music* and *no_music*, but tends to fail when it comes to reliably detecting music within mixed segments containing for example speech and music. (We checked that by reimplementing their feature in our framework.) A deeper analysis of the feature led to the conclusion that concentrating on absolut energy values of the

spectogram has a counter-productive effect, because the horizontal bars might be rather soft and the absolute values of foreground sounds will have a stronger impact. Keum et al. [12] recently introduced a feature that relies on a binarization step to neglect the absolute strength of an activation. However, their binarization threshold is chosen so as to remove the small magnitudes, which is equivalent to removing all the soft activations corresponding to the soft musical tones we want to detect. In the next section a new feature is proposed to make the detection of continuous frequency activations more reliable, even if other audio signals are present simultaneously.

5.2. The feature extraction process

The computation of the *Continuous Frequency Activation* (CFA) of an audio stream can be subdivided into several steps:

- **Conversion of the input audio stream**
  The input stream is converted to 11 kHz and mono.
- **Computation of the power spectrum**
  We compute the power spectrum using a Hanning window function and a window size of 1024 samples, corresponding to approximately 100ms of audio. A hop-size of 256 samples is used, resulting in an overlap of 75% percent. After the conversion to decibel, we obtain a standard spectrogram representation.
- **Emphasize local peaks**
  To emphasize local energy peaks within each frame of the STFT we subtract from the power spectrum of each frame the running average using a window size of $N = 21$ frequency bins:

$$X_i^{emph} = X_i - \frac{1}{N} \sum_{k=-\frac{N}{2}}^{\frac{N}{2}} X_{\min(\max(k,1),N)} \qquad (6)$$

  where X_i denotes the energy of the i-th frequency component of the current frame. This step is useful to emphasize very soft tones, belonging to background music: if a soft tone is not masked by another signal over its entire duration (which is unlikely, as non-music signals tend to be less stationary), the perceivable horizontal bars in the spectogram are compositions of consecutive local maxima. Thus, we try to emphasize these soft bars by emphasizing all local maxima in the spectrum of a frame.
- **Binarization**
  To neglect the absolute strength of activation (energy) in a given frame j, we binarize each frequency component X_{ij}^{emph} by comparing to a fixed binarization threshold. The binarization threshold $t = 0.1$ was chosen in such a way that even soft activations could be kept in the binarized spectogram. Only frequency bins which are obviously not active at all, will be set to 0 using this low threshold. This is an important difference to Keum et al. [12], who apply a threshold to remove small magnitudes.

$$B_{ij} = \begin{cases} 1 & X_{ij}^{emph} > t \\ 0 & X_{ij}^{emph} \leq t \end{cases} \qquad (7)$$

  Neglecting the actual strength of the activation allows us to focus on structural aspects of the emphasized spectrogram only.
- **Computation of the frequency activation**
  We further process the binarized power spectrum in terms of *blocks*. Each block consists of $F = 100$ frames and blocks overlap by 50%, which means that a block is an excerpt of the binarized spectrogram corresponding to 2.6 seconds of audio. For each block we compute the frequency activation function $Activation(i)$. For each frequency bin i, the frequency activation function measures how often a frequency component is active in a block. We obtain the frequency activation function for a block by simply summing up the binarized values for each frequency bin i:

$$Activation(i) = \frac{1}{F} \sum_{j=1}^{F} B_{ij} \qquad (8)$$

  Normalizing the frequency activation by the length of the block is not necessary, but would make it possible to compare results from different block lengths. Figure 2 shows the binarized emphasized power spectra of two blocks and the resulting frequency activation functions. Subplot **(b)** is typical of blocks containing music, whereas subplot **(a)** is representative for blocks without any musical elements.
- **Detect strong peaks**
  Strongs peaks in the frequency activation function of a given bock indicate steady activations of narrow frequency bands. The "spikier" the activation function, the more likely horizontal bars, which are characteristic of sustained musical tones, are present. Even one large peak is quite a good indicator for the presence of a tone. The peakiness of the frequency activation function is consequently a good indicator for the presence of music. To extract the peaks we use the following simple peak picking algorithm.

  1. Collect all local peaks, starting from the lowest frequency. Each local maximum of the activation function is a potential peak (and there are many of them – cf. Figure 2).
  2. For each peak x_p, compute its height-to-width index or *peak value* $pv(x_p) = h(x_p)/w(x_p)$, where the height $h(x_p)$ is defined as $\min[f(p) - f(x_l), f(p) - f(x_r)]$, with $f(x)$ the value of the activation function at point (frequency bin) x and x_l and x_r are closest local minima of f to the left and right of x_p, respectively. The width $w(x_p)$ of the peak is given by:

$$w(x_p) = \begin{cases} p - x_l & f(p) - f(x_l) < f(p) - f(x_r) \\ x_r - p & otherwise \end{cases}$$

  Steps 1 and 2 can be done in one left-to-right scan of the activation function.
- **Quantify the Continuous Frequency Activation**
  To quantify the *Continuous Frequency Activation* of the activation function of a block, the pv values of all detected peaks are sorted in descending order, and the sum of the five largest peak values is taken to characterize the overall "peakiness" of the activation function.

As a result of this lengthy extraction process we obtain exactly one numeric value for each block of frames, which quantifies the presence of steady frequency components within the current audio segment. For blocks containing music the resulting value should be higher than for blocks where no music is present.

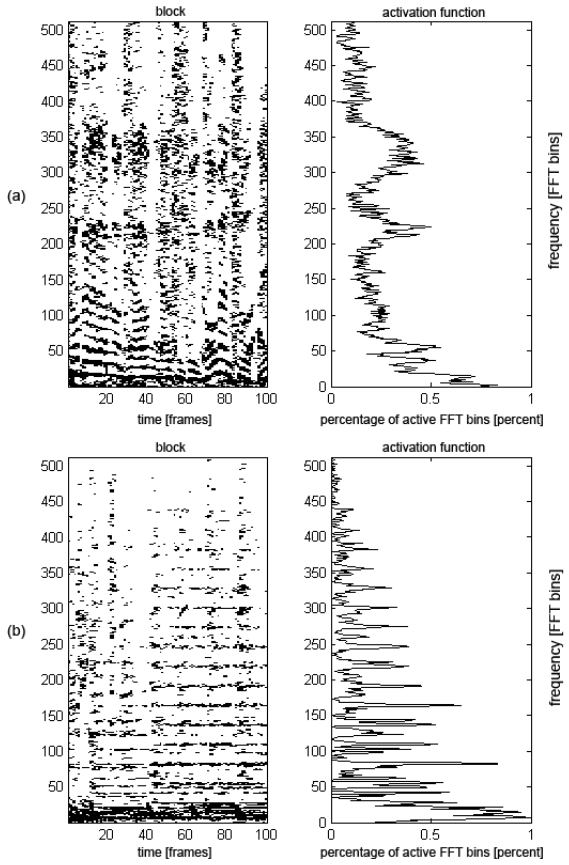


Figure 2: *Binarized spectogram of a block and the corresponding activation function. Block **(a)** contains no music, whereas in block **(b)** music is present.*

5.3. Results using the new feature

Returning just a single numeric value, the newly proposed feature simplifies the classification process a lot. The separation of the two classes *music* and *no_music* can be done by a simple comparison with a threshold t. Optimising the threshold on our training set (the top 3 shows in Table 1) yielded a value of $t = 1.24$.

Table 5 shows the percentage predictions on the test set with this threshold value, after the application of the smoothing function introduced in section 3.3. Only two estimates, highlighted in bold face, exceed the error level of 10 percentage points. To illustrate the effectiveness of the CFA, Figure 5 once more shows the automatic segmentation of *"Julia"*. It is clearly visible that the CFA feature makes far fewer mistakes even before smoothing. The classification accuracy of **81.21%** obtained with the machine learning approach improves to **89.93%**, although now just one feature and simple thresholding is used. This compares favorably to the 75.86% we reconstructed from the results reported by [1] on a related problem (see section 2 above).

Figure 3 compares the real percentages of music present, the percentages predicted by the machine learning approach, and the percentages estimated using CFA alone. There are still some cases where the CFA feature fails. Especially when the continuous frequency activations are a consequence of continuous noise signals, such as e.g. helicopter noise, the CFA feature wrongly detects music segments. Again, some examples of those misclassifications can be found at (`http://www.cp.jku.at/people/seyerlehner/md.html`).

We also tested the CFA feature on a different set of reference data, namely, the Scheirer-Slaney database [13], which consists of 245 samples of radio recordings (which are presumably easier to classify than our data). To our knowledge, this is currently the only dataset of this kind that is publicly available.[2] The dataset was split into a training and a test set by Dan Ellis and is described in detail in [15]. The only change we made was to reduce the classes to *music* and *no_music* only. Based on this training set of 184 examples, we found an empirical threshold of $t = 1.05$. Using this threshold, 60 out of the 61 examples of the test set were classified correctly – a classification accuracy of 98.36%, which is

title	% real	% est.	diff.
Alpen Donau Adria	57.08 %	48.61 %	8.47
Barbara Karlich Show	7.51 %	6.64 %	0.87
Da wo es noch Treue gibt	62.90 %	63.50 %	0.60
Frisch gekocht	10.01 %	6.69 %	3.32
Gut beraten Österreich	8.76 %	5.74 %	3.02
Heilige Orte	54.34 %	42.70 %	**11.64**
Heimat fremde Heimat	29.72 %	17.33 %	**12.39**
Hohes Haus	17.50 %	9.26 %	8.24
Julia	80.36 %	76.88 %	3.48
ZIB	4.91 %	0 %	4.91

Table 5: *The percentage of music really present versus the percentage estimated using Continuous Frequency Activation.*

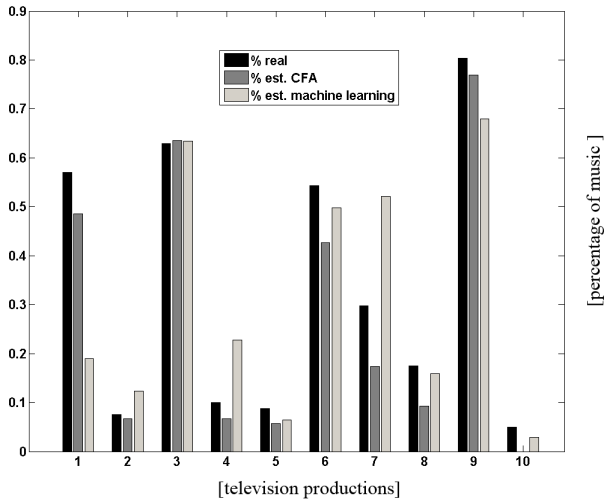


Figure 3: *Comparison of the real percentage of music, the machine learning estimate and the CFA estimate (see tables 4 and 5; The numbering of the television productions corresponds to the rows in these tables.)*

[2] We hope to get the permission by the Austrian National Broadcasting Corporation (ORF) to make our ground truth data available online.

roughly comparable to the results reported in [14].

6. CONCLUSIONS

In this paper we introduced a new music detection application, namely music detection in TV productions, and pointed out that this application differs from common speech/music classification problems. Our experiments show that standard speech/music discrimination features in combination with standard machine learning algorithms yield interesting, but highly varying results. Therefore we focused on the development of more reliable features.

Extending standard frame-level features to clip-level features, thus incorporating some rudimentary temporal information, seems not to be a successful strategy. On the other hand, our experiments show that the application of an appropriate smoothing function results in plausible segmentations and improves the overall accuracy considerably.

We then introduced a new feature which was especially designed to detect music in an accurate and robust way. This raised the total accuracy on our highly non-trivial test set to **89.93%**. Surprisingly, a simple thresholding approach based on this new feature alone outperforms the machine learning approach. This supports the thesis that music detection can be further improved if one makes use of the structural aspects of music, which even allows the detection of background music.

We have plans to further optimize the parameter settings of the *Continuous Frequency Activation* and to develop other features exploiting structural properties of music signals. One interesting direction might be to focus on rhythmic properties, as Scheirer and Slaney [13] and Jarina et al. [16] have already tried. With respect to the current application, we will also investigate combinations of speech/music discrimination features and the CFA feature and hope to deploy an operational music detection system at the Austrian National Broadcasting Corporation (ORF) in the near future.

7. ACKNOWLEDGMENTS

We would like to thank Harald Frostel and Richard Vogl, who implemented large parts of the experimentation framework built around the `WEKA`[10] machine learning environment. Thanks to Dan Ellis for making his dataset available. This research was supported in part by the Austrian *Fonds zur Förderung der Wissenschaftlichen Forschung (FWF)* under grant L112-N04.

8. REFERENCES

[1] M.D. Santo, G. Percannella, C. Sansone, and M. Vento, "Classifying audio of movies by a multi-expert system," in *Proc. of the 11th International Conference on Image Analysis and Processing (ICIAO'01), Palermo, Italy*, 2001, pp. 386–392.

[2] M.K.S. Khan, W.G. AI-Khatib, and M. Moinuddin, "Classifying audio of movies by a multi-expert system," in *Proc. Proceedings of the 2nd ACM international workshop on Multimedia databases (MMDB'04), Washington, DC, USA*, 2001, pp. 386–392.

[3] K. Minami, A. Akutsu, H. Hamada, and Y. Tonomura, "Video handling with music and speech detection," *IEEE MultiMedia*, vol. 5, no. 3, pp. 17–25, 1998.

[4] K. Minami, A. Akutsu, H. Hamada, and Y. Tonomura, "Enhanced video handling based on audio analysis," in *Proc. of the 1997 International Conference on Multimedia Computing and Systems (ICMCS'97), Washington, DC, USA*, 1997, pp. 219–226.

[5] J. Mauclair and J. Pinquier, "Fusion of descriptors for speech / music classification," in *Proc. of the 12th European Signal Processing Conference (EUSIPCO'04), Vienna, Austria*, 2004.

[6] H. Misra, S. Ikbal, H. Bourlard, and H. Hermansky, "Spectral entropy based feature for robust asr," in *Proc. of of the 29th International Conference on Acoustics, Speech, and Signal Processing (ICASSP'04), Montreal, Canada*, 2004, pp. 193–196.

[7] A. Pikrakis, T. Giannakopoulos, and S. Theodoridis, "A computationally efficient speech/music discriminator for radio recordings," in *Proc. of of the 7th International Conference on Music Information Retrieval (ISMIR'06), Victoria, Canada*, 2006, pp. 107–110.

[8] H. Ezzaidi and J. Rouat, "Speech, music and songs discrimination in the context of handsets variability," in *Proc. of of the 7th International Conference on Spoken Language Processing (ICSLP'02), Denver, USA*, 2002, pp. 16–20.

[9] M.K.S. Khan and W.G. Al-Khatib, "Machine-learning based classification of speech and music," *Multimedia Systems*, vol. 12, no. 1, pp. 55–67, 2006.

[10] I.H. Witten and E. Frank, *Data Mining: Practical machine learning tools and techniques, 2nd Edition*, Morgan Kaufmann, 2005.

[11] M. Hawley, *Structure Out of Sound*, Ph.D. thesis, Massachusetts Institute of Technology. Dept. of Architecture. Program in Media Arts and Sciences, 1993.

[12] J. Keum and H. Lee, "Speech/music discrimination based on spectral peak analysis and multi-layer perceptron," in *Proc. of the 9th International Conference on Hybrid Information Technology (ICHIT'06), Cheju Island, Korea*, 2006, pp. 56–61.

[13] E. Scheirer and M. Slaney, "Construction and evaluation of a robust multifeature speech/music discriminator," in *Proc. of the 22th International Conference on Acoustics, Speech, and Signal Processing (ICASSP'97), Munich, Germany*, 1997, pp. 1331–1334.

[14] G. Williams and D. Ellis, "Speech/music discrimination based on posterior probability features," in *Proc. of the 6th European Conference on Speech Communication and Technology (EUROSPEECH'99), Budapest, Hungary*, 1999, pp. 687–690.

[15] D.P.W. Ellis, "The Music-Speech Corpus," Available at http://labrosa.ee.columbia.edu/sounds/musp/scheislan.html, Accessed March 21, 2007.

[16] R. Jarina, N.O. Connor, S. Marlow, and N. Murphy, "Rhythm detection for speech-music discrimination in mpeg compressed domain," in *Proc. of the 14th International Conference on Digital Signal Processing (DSP'02), Hellas, Greece*, 2002, pp. 129–132.

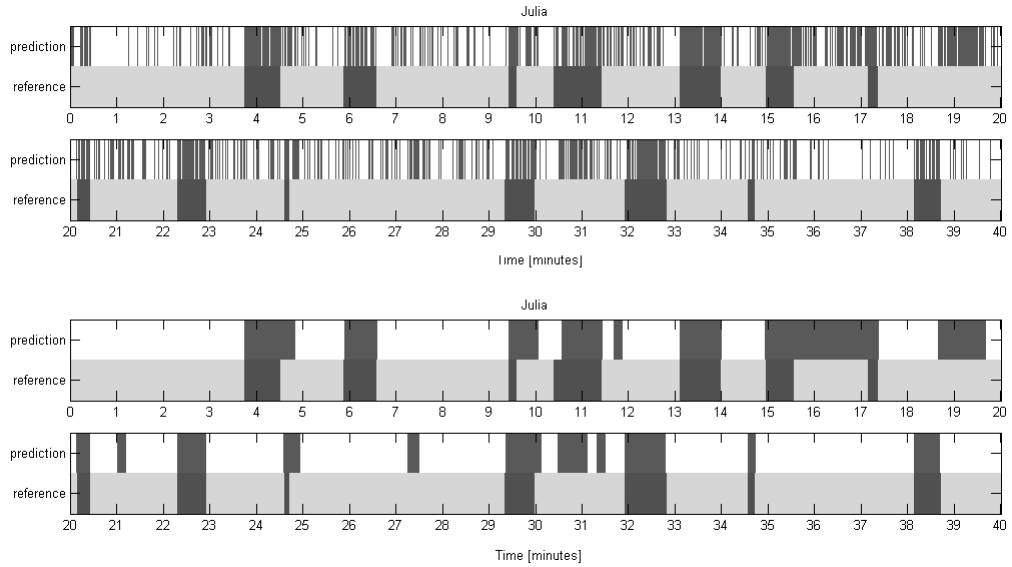


Figure 4: *Visualization of the classification results for the **machine learning approach** of 40 minutes of the soap opera "Julia". Each line represents 20 minutes of audio and is split to compare the class prediction with the true class. The class "music" is represented by a lighter color, whereas "no_music" is in the form of dark regions. The lower subplot illustrates the results after the application of the smoothing function.*

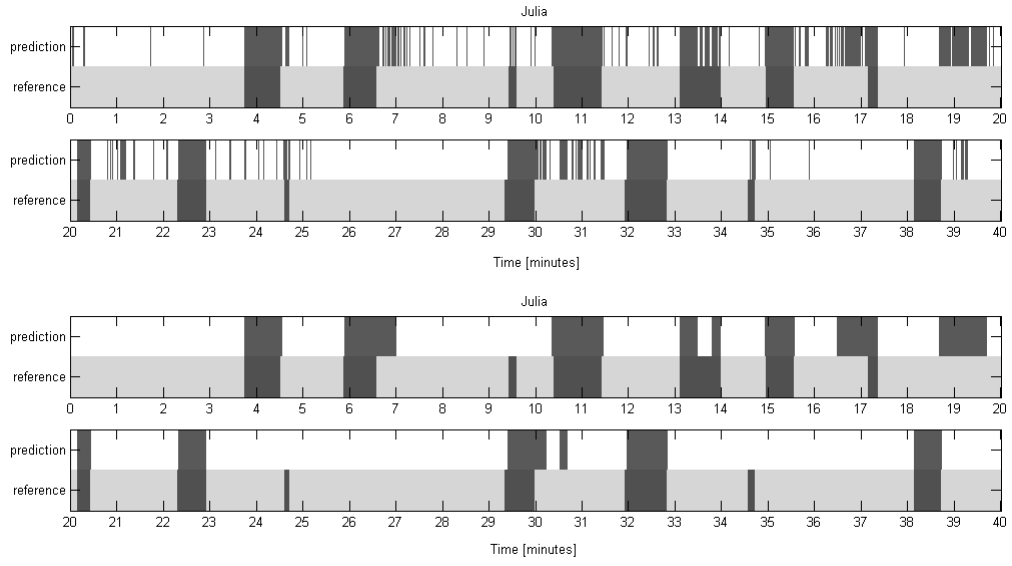


Figure 5: *Visualization the classification results for the new **CFA** feature of 40 minutes of the soap opera "Julia". Each line represents 20 minutes of audio and is split to compare the class prediction with the true class. The class "music" is represented by a lighter color, whereas "no_music" is shown in the form of dark regions. The lower subplot illustrates the results after the application of the smoothing function.*

CHORUS DETECTION WITH COMBINED USE OF MFCC AND CHROMA FEATURES AND IMAGE PROCESSING FILTERS

Antti Eronen
Nokia Research Center
Tampere, Finland
Antti.Eronen@nokia.com

ABSTRACT

A computationally efficient method for detecting a chorus section in popular and rock music is presented. The method utilizes a distance matrix representation that is obtained by summing two separate distance matrices calculated using the mel-frequency cepstral coefficient and pitch chroma features. The benefit of computing two separate distance matrices is that different enhancement operations can be applied on each. An enhancement operation is found beneficial only for the chroma distance matrix. This is followed by detection of the off-diagonal segments of small distance from the distance matrix. From the detected segments, an initial chorus section is selected using a scoring mechanism utilizing several heuristics, and subjected to further processing. This further processing involves using image processing filters in a neighborhood of the distance matrix surrounding the initial chorus section. The final position and length of the chorus is selected based on the filtering results. On a database of 206 popular & rock music pieces an average F-measure of 86% is obtained. It takes about ten seconds to process a song with an average duration of three to four minutes on a Windows XP computer with a 2.8 GHz Intel Xeon processor.

1. INTRODUCTION

Music thumbnailing refers to the extraction of a characteristic, representative excerpt from a music file. Often the chorus or refrain is the most representative and "catchiest" part of a song. A basic application is to use this excerpt for previewing a music track. This is very useful if the user wishes to quickly get an impression of the content of a playlist, for example, or quickly browse the songs in an unknown album. In addition, the chorus part of a song would often make a good ring tone for a mobile phone, and automatic analysis of the chorus section would thus facilitate extraction of ring tone sections from music files.

Western popular music is well suited for automatic thumbnailing as it often consists of distinguishable sections, such as intro, verse, chorus, bridge, and outro. For example, the structure of a song may be intro, verse, chorus, verse, chorus, chorus. Some songs do not have as clear verse-chorus structure but there still are separate sections, such as section A and section B that repeat. In this case the most often repeating and energetic section is likely to contain the most recognizable part of the song.

Peeters et al. ([1]) divide the methods for chorus detection and music structure analysis into two main categories: the "state approach" which is based on clustering feature vectors to states having distinctive statistics, and the "sequence approach" which is based on computing a self-similarity matrix for the signal. One of the first examples of the state approach was that of Logan and Chu [2]. Recently, e.g. Levy et al. [3] and Rhodes et al. [4] have studied this approach. Similarity-matrix based approaches include the ones by Wellhausen & Crysandt [5] and Cooper & Foote [6]. Bartsch & Wakefield [7] and Goto [8] operated on an equivalent time-lag triangle representation. There are also methods utilizing many different cues, including e.g. segmentation into vocal / nonvocal sections, such as [9], or methods that iteratively try to find an optimal segmentation [10].

Here we present a method for detecting the chorus or some other often repeating and representative section from music files. The method is based on the self-similarity (distance) representation. The goal was to device a computationally efficient method that still would produce high quality music thumbnails for practical applications. Thus, iterative methods based on feature clustering or computationally intensive optimization steps could not be used. The following summarizes the novel aspects of the proposed method:

The self-distance matrix (SDM) used in the system is obtained by summing two distance matrices calculated using MFCC and chroma features. This improves the performance compared to the case when either of the features would be used alone. Although the MFCC features are sensitive to changing instrumentation between the occurrences of the chorus, the fact that the instrumentation and expression during the chorus is often different than in other parts of the song seems to overweigh this, at least with our pop & rock dominated data. The benefit of the proposed distance-matrix summing approach instead of merely concatenating the features into one, longer vector is that different enhancement operations can be applied for each matrix.

An initial chorus section is obtained from the repetitions detected from the SDM by utilizing a novel heuristic scoring scheme. The heuristics consider aspects such as the position of a repetition in the self-distance matrix (SDM), the adjustment of a repetition in relation to other repetitions in the SDM, average energy and average distance in the SDM during the repetition, and number of times the repetition occurs in the musical data.

The system performs the chorus determination in two steps: first a preliminary candidate is found for the chorus section, and then its final location and duration is determined by filtering with a set of image processing filters, selecting the final chorus position and duration according to the filter which gives the best fit.

Evaluations are presented on a database of 206 popular and rock music pieces. The method is demonstrated to provide sufficient accuracy for practical applications while being computationally efficient.

2. METHOD

2.1. Overview

Figure 1 shows an overview of the proposed method, which consists of the following steps. First the beats of the music signal are detected. Then, beat synchronous mel-frequency cepstral coefficient (MFCC) and pitch chroma features are calculated. This results in a sequence of MFCC and chroma feature vectors. Next, two self-distance matrices (SDM) are calculated, one for the MFCC features and one for the chroma features. Each item in the SDM represents the distance of feature vector at beat *i* to a feature vector at beat *j*. In the distance matrix representation, choruses or other repeating sections are shown as diagonal lines of small distance. The diagonal lines of the chroma distance matrix are then enhanced. Next we obtain a summed distance matrix by summing the chroma and MFCC distance matrices. This is followed by binarization of the summed distance matrix, which attempts to detect the diagonal regions of small distance (or high similarity). From the detected diagonal segments, the most likely chorus section candidate (diagonal stripe) is selected, and subjected to further processing. This further processing involves using image processing filters in a neighbourhood of the similarity matrix which surrounds the most likely chorus candidate. The final position and length of the chorus is selected based on the image processing results.

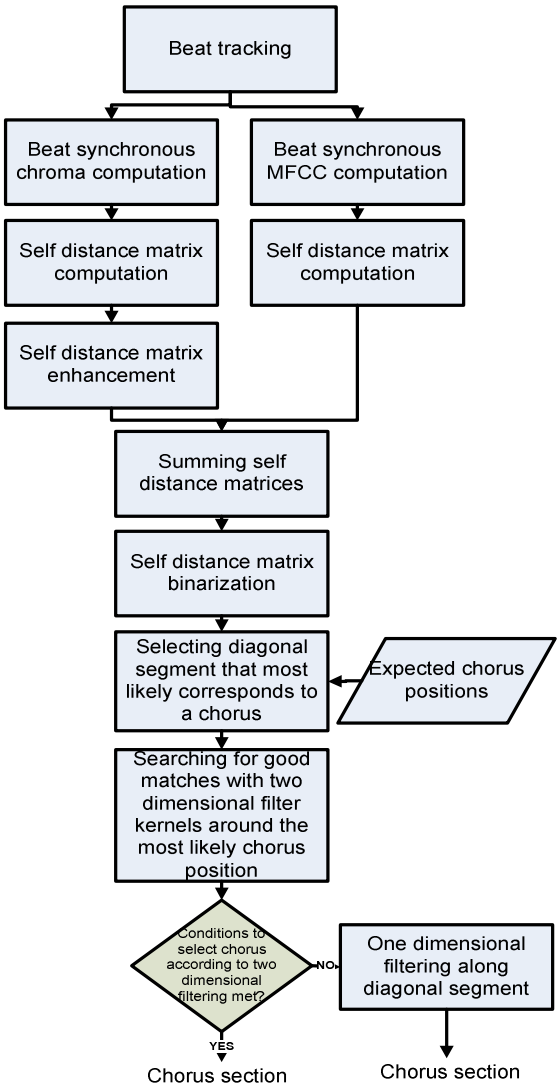


Figure 1: *Overview of the proposed method.*

2.2. Beat tracking

The feature extraction step begins by finding the beats in the acoustic music signal. We utilize the efficient beat tracking method described in [11] to produce an initial set of beat times and an accent signal $v(n)$. The accent signal measures the change in the spectrum of the signal and exhibits peaks at onset locations. An additional, non-causal postprocessing step was implemented to prevent the beat interval from changing significantly from one frame to another, which might cause problems with the beat synchronous self-distance matrices. The postprocessing is performed with a dynamic programming method described by Ellis [12]. The dynamic programming step takes as input the accent signal and median beat period produced by the method described in [11], performs smoothing of the accent signal with a Gaussian window, and then finds the optimal sequence of beats through the smoothed accent signal. The method iterates through each sample of the smoothed accent signal, and finds the best previous beat time for each time sample. The selection is affected by the strength of the accent signal at the previous beat position, and the difference to the ideal beat interval. The indices of best previous beats are stored for each time sample, and in the end the single best sequence is obtained by backtracking through the previous beat records. For more details see [12].

2.3. Feature calculation

Next, beat synchronous MFCC and chroma features are calculated. Analysis frames are synchronized to start at a beat time and end before the next beat time, and one feature vector for each beat is obtained as the average of feature values during that beat. Beat synchronous frame segmentation has been used earlier e.g. in [7]. It has two main advantages: it makes the system insensitive to tempo changes between different chorus performances, and significantly reduces the size of the self-distance matrices and thus computational load. Prior to the analysis, the input signal is downsampled to 22050 kHz sampling rate.

The MFCC features are calculated in 30 ms hamming windowed frames during each beat, and the average of 12 MFCC features (ignoring the zeroth coefficient) for each beat is stored. We use 36 frequency bands spaced evenly on the mel-frequency scale, and the filters span the frequency range from 30Hz to the nyquist frequency. Chroma features are calculated in longer, 186 ms frames to get a sufficient frequency resolution for the lower notes. In our implementation, each bin of the discrete Fourier transform is mapped to exactly one of the twelve pitch classes C, C#, D, D#, E, F, F#, G, G#, A, A#, B, with no overlap. The energy is calculated from a range of six octaves from C3 to B8 and summed to the corresponding pitch classes. The chroma vectors are normalized by dividing each vector by its maximum value.

After the analysis, each inter-beat interval is represented with a MFCC vector and chroma vector, both of which are 12-dimensional.

2.4. Distance matrix calculation

The next step is to calculate the self-distance matrix (SDM) for the signal. Each entry $D(i, j)$ in the distance matrix indicates the distance of the music signal at time i to itself at time j. As we are using beat synchronous features, time is measured in beat units. Two distance matrices are used, one for the MFCC features and one for the chroma features. The entry $D_{mfcc}(i, j)$ of the MFCC distance matrix is calculated as the Euclidean distance of MFCC vectors of beats i and j. Correspondingly, in the chroma distance matrix $D_{chroma}(i, j)$ each entry corresponds to the Euclidean distance of the chroma vectors of beats i and j. Figures 2 and 3 show examples of a chroma and MFCC distance matrices, respectively. As the Euclidean distance is symmetric, the distance matrix will also be symmetric. Thus, the following operations consider only the lower triangular part of the distance matrix.

Alternatives to calculating two different distance matrices would be to concatenate the features before calculating the distances, or combine the features in the distance calculation step. The benefit of keeping the distance matrices separate is that different enhancement operations can be applied to the chroma and MFCC matrices. Based on our experiments, it seems beneficial to apply an enhancement only for the chroma distance matrix and not for the MFCC distance matrix. When long chords or notes are played during several adjacent beats, the chroma distance matrix will exhibit a square area of small distance values. An enhancement operation similar to the one described in [8] was found to be beneficial in removing these. The MFCC distance matrix does not exhibit similar areas as the MFCC features are insensitive to pitch information, so this would explain the MFCC distance matrix does not benefit from the enhancement. Moreover, summing the distance matrices first and then enhancing the summed matrix did not perform as well as enhancing the chroma matrix only and then summing with the MFCC matrix. The next section describes the used enhancement and SDM summing steps.

2.5. Enhancing and summing the distance matrices

Ideally, the distance matrix should contain diagonal stripes of small distance values at positions corresponding to repetitions of the chorus or refrain section. However, due to variations in the performance of the chorus at different times (articulation, improvisation, changing instrumentation), the diagonal stripes are often not very well pronounced. In addition, there may be additional small distance regions which do not correspond to chorus sections. To make diagonal segments of small distance values more pronounced in the distance matrix, an enhancement method similar to the one presented in [8] is utilized.

The chroma distance matrix $D_{chroma}(i, j)$ is processed with a 5 by 5 kernel. For each point (i, j) in the chroma distance matrix, the kernel is centred to the point (i, j). Six directional local mean values are calculated along the upper-left, lower-right, right, left, upper, and lower dimensions of the kernel. If either of the means along the diagonal is the minimum of the local mean values, the point (i, j) in the distance matrix is emphasized by adding the minimum value. If some of the mean values along the horizontal or vertical directions is the smallest, it is assumed that the value at (i, j) is noisy and it is suppressed by adding the largest of the local mean values. After the enhancement the diagonal lines corresponding to repeating sections are more pronounced.

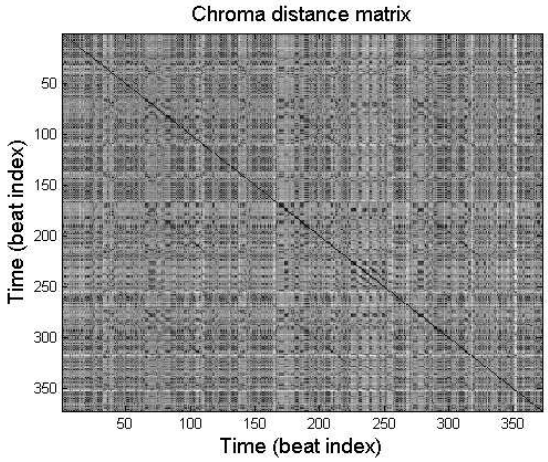


Figure 2: *The chroma distance matrix $D_{chroma}(i, j)$ of the song "Like a virgin" by Madonna.*

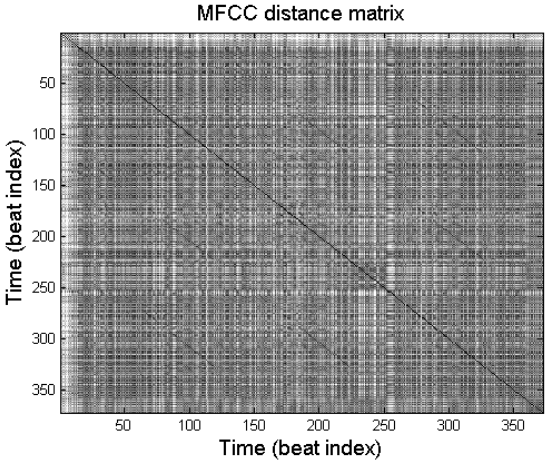


Figure 3: *The MFCC (timbre) distance matrix $D_{mfcc}(i, j)$ of the song "Like a virgin" by Madonna.*

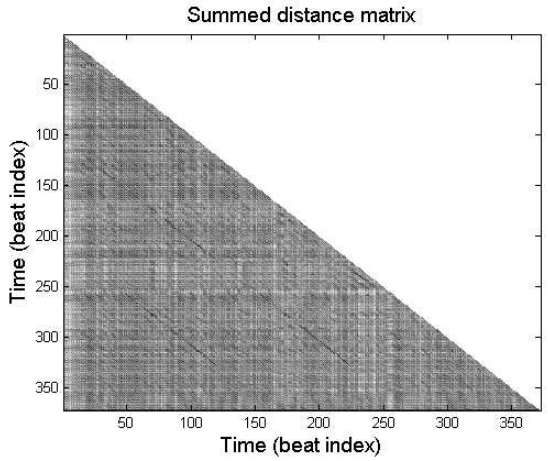


Figure 4. *The final distance matrix $D(i, j)$ of the song "Like a virgin" by Madonna obtained after summing the enhanced chroma distance matrix and MFCC distance matrix.*

After the enhancement step the chroma and MFCC distance matrices are summed. This gives the final distance matrix *D*, where the entries $D(i,j) = \widetilde{D}_{chroma}(i,j) + D_{mfcc}(i,j)$, where $\widetilde{D}_{chroma}$ is the chroma distance matrix after the above described enhancement operation. Figure 4 shows the summed distance matrix for Madonna's "Like a virgin". Weighted summation was also attempted for the different matrices with certain weight combinations, but equal weights (i.e. no weighting) seem to perform well. A slightly related approach to our distance matrix summing was presented by Marolt [13]. He constructed several beat synchronous melodic representations by comparing excerpts of different length, and then combined the representations by pointwise multiplication. This was reported to help in reducing noise in the self-similarity representation.

2.6. Detecting repetitions from the self-distance matrix

The following step attempts to detect which parts of the distance matrix correspond to a repetitive segment and which do not. The binarization method used here is similar to the one presented by Goto in [8], except that we operate on the low-triangular part of a distance matrix whereas Goto operated on the time-lag triangle. In addition, the filtering operations are simplified here and the threshold selection operations differ slightly.

When a sum is calculated along a diagonal segment of the distance matrix, a smaller value indicates a larger likelihood that the particular diagonal contains one or more line segments with small similarity values. A sum is calculated along the low-left diagonals *k* of the distance matrix, giving the values

$$F(k) = \frac{1}{M-k}\sum_{c=1}^{M-k} D(c+k,c)\ ,\ k = 1,...,M-1 \qquad (1)$$

where M is the number of beats in the song. Thus, $F(1)$ corresponds to the first diagonal below the main, $F(2)$ to the second below the main diagonal, and so on. The values of *k* corresponding to the smallest values of *F*(*k*) indicate diagonals which are likely to have repetitions in them. With Eq. 1 there exists a possibility that some small-distance values are masked by high distance values that happen to locate at the same diagonal. Thus, it might be worth studying whether special methods to remove the effect of high-distance values would improve the performance. However, this was left for future research as the simple summing seems to work well.

A certain number of diagonals corresponding to minima in *F*(*k*) are then selected. Before looking for minima in *F*(*k*), it is "detrended" to remove cumulative noise from it. This is done by calculating a lowpass filtered version of *F*(*k*), using a FIR lowpass filter with 50 taps, the value of each coefficient being 1/50. The lowpass filtered version of *F*(*k*) is subtracted from *F*(*k*).

The minima correspond to zero-crossings in the differential of *F(k)*. The smoothed differential of *F*(*k*) is calculated by filtering *F*(*k*) with an FIR filter having the coefficients $b_1(i) = K-i$, $i = 0, \ldots, 2K$, with $K = 1$. The minima candidates are obtained by finding the points where the smoothed differential of *F*(*k*) changes its sign from negative to positive. The values of the minima are dichotomized into two classes with the Otsu method presented in [14], and the values smaller than the threshold are selected. We observed that sometimes it may happen that only a few negative peaks are selected using this threshold. This would mean that the following binarization would examine only a few diagonals of the distance matrix, increasing the possibility that some essential diagonal stripes are left unnoticed. To overcome this, we raise the threshold gradually until at least 10 minima (and thus diagonals) are selected. The subset of indices selected from all the diagonal indices $k \in [1, M-1]$ to search for line segments is denoted by *Y*.

The diagonals of the SDM selected for the line segment search are denoted by

$$g_y(c) = D(c+y,c)\ ,\ c = 1,...,M-y \qquad (2)$$

where $y \in Y$. The diagonals $g_y(c)$ of the distance matrix are smoothed by filtering with a FIR with coefficients $b_2(i) = ¼$, $i = 1, \ldots, 4$. Goto ([8]) performed another threshold selection with the Otsu method ([14]) to select a threshold to be used for detecting the line segments from the diagonals. However, we found it better to define a threshold such that 20% of the values of the smoothed diagonals $\widetilde{g}_y(c)$ are left below it, and thus 20% of values are set to correspond to diagonal repetitive segments. This threshold is obtained in a straightforward manner by concatenating all the values of $\widetilde{g}_y(c)$, $c = 1,...,M-y$ and $y \in Y$ into a long vector, sorting the vector, and selecting the value such that 20 % of the values are smaller. Points where $\widetilde{g}_y(c)$ exceeds the threshold are then set to one, others are set to zero. This gives the binarized distance matrix.

Next the binarized matrix is enhanced, such that diagonal segments where most values are ones (i.e. detected small distance segments) are enhanced to be all ones under certain conditions. This is done in order to remove gaps of few beats in such diagonal segments that are long enough. These kinds of gaps occur if there is a point of high distance within a diagonal segment (due to e.g. a variation in the musicians' performance). The enhancement process processes the binarized distance matrix with a kernel of length 25 (beats). Thus, at the position (*i*, *j*) of the binarized distance matrix B(*i*, *j*), the kernel analyzes the diagonal segment from *B*(*i*, *j*) to *B*(*i*+25-1, *j*+25-1). If at least 65 % of the values of the diagonal segment are ones, *B*(*i*, *j*) = 1 and either *B*(*i*+25-2, *j*+25-2) = 1 or *B*(*i*+25-1, *j*+25-1) = 1, all the values in the segment are set to one. This removes short gaps in the diagonal segments. The length of the kernel is a parameter to the system, the value 25 was found to work well. Goto ([8]) did not report a need for such an enhancement process but we found it necessary.

2.7. Locating interesting segments

The result of the previous steps is an enhanced binarized matrix $B_e(i,j)$ where the value one indicates that that point corresponds to a repetitive section and zero corresponds that there is no repetition at that point. The next step is to find diagonal segments that are interesting, i.e. likely correspond to a chorus.

There may be repetitions that are too short to correspond to a chorus, such as those that occur e.g. when the same pattern of notes are repeatedly played with some instrument. By default, segments longer than four seconds are searched and used for further processing. In the case no segments longer than four seconds are found, the system tries to extend the segments until at least some segments longer than four seconds are detected. If this does not help, the length limit is relaxed and all segments are used.

With some songs there may be a very large number of repetitive diagonal segments at this point. Therefore, some of the segments are removed. For each diagonal segment found in the binarized matrix, the method looks for diagonal segments which are located close to it. Let us denote a diagonal segment which starts at (i, j) and ends at (i', j') with $\underline{x}_p = [i, j, i', j']$. Furthermore, the length $\Delta(\underline{x}_p) = j' - j + 1$ is the duration of the segment in beats. Given two segments $\underline{x}_1$ and $\underline{x}_2$, the segment $\underline{x}_2$ is defined to be close to $\underline{x}_1$ iff

$\underline{x}_2(1) \geq (\underline{x}_1(1) - 5)$ and $\underline{x}_2(3) \leq (\underline{x}_1(3) + 20)$ and

$|\underline{x}_2(2) - \underline{x}_1(2)| \leq 20$ and $\underline{x}_2(4) \leq (\underline{x}_1(4) + 5)$

where |.| denotes absolute value. The parameters were obtained by experimentation and may be changed.

For each segment, the method then lists its close segments fulfilling the conditions above, finds the segments that have more than three close segments, and removes the extra segments. If some segment with more than three close segments is in the removal list of some other segment, then it is not removed. The result of this step is a collection of the diagonal segments $\underline{x}_p$, $p = 1,...,P$ in the binarized matrix.

2.8. Selecting the diagonal segment most likely corresponding to a chorus

Next the method selects the segment most likely corresponding to a chorus. This is done by utilizing a novel heuristic scoring scheme which considers aspects such as the position of a repetition in the self distance matrix, the position of a repetition in relation to other repetitions in the SDM, average energy and average distance in the SDM during the repetition, and number of times the repetition occurs in the musical data.

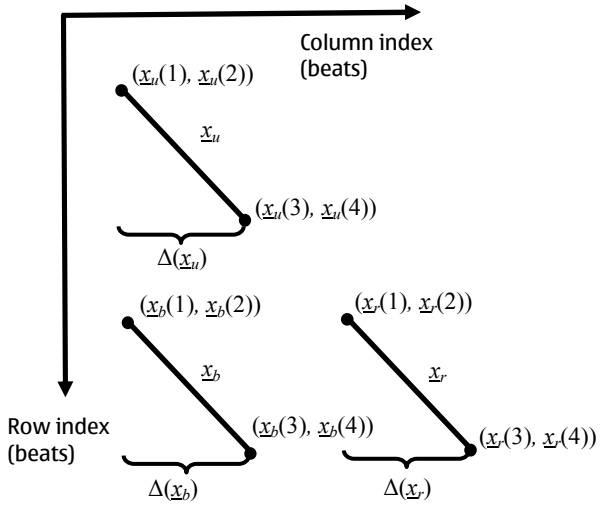


Figure 5: *Notations when giving scores to a group of three diagonal segments (detected stripes of small distance of the distance matrix). The units are measured in beats.*

2.8.1. Position of a repetition in the distance matrix

The first criterion used in making the decision is how close a diagonal segment is to an expected chorus position in the song. This is based on the observation that often in pop music there is a chorus at time corresponding to approximately one quarter of the song length. A partial score $s_1(\underline{x}_p)$ measures the difference of the middle column of segment $\underline{x}_p = [i, j, i', j']$ to one quarter of the song length:

$$s_1(\underline{x}_p) = 1 - \frac{\left|(j + \Delta(\underline{x}_p)/2) - round(M/4)\right|}{round(M/4)}, \quad (3)$$

where *M* is the length of the song in beats. The partial score $s_2(\underline{x}_p)$ measures the difference of the middle row of segment $\underline{x}_p$ to three quarters of the song length:

$$s_2(\underline{x}_p) = 1 - \frac{\left|(i + \Delta(\underline{x}_p)/2) - round(3 \cdot M/4)\right|}{round(M/4)}. \quad (4)$$

With $s_1(\underline{x}_p)$ and $s_2(\underline{x}_p)$ we give more weight to such segments that are close to the position of the diagonal stripe on the low left hand corner of Figure 4, which corresponds to the first occurrence of a chorus (and match to the third occurrence) and is often the most prototypically performed chorus, i.e. no articulation or expression.

2.8.2. Adjustment in relation to other repetitions

The second criterion relates to the adjustment of a segment within the distance matrix in relation to other repetitions. Motivated by the approach presented in [5], we look for possible groups of three diagonal stripes that might correspond to three repetitions of the chorus. See Figure 5 for an example of an ideal case. The search for possible groups of three stripes is done as follows: the method goes through each found diagonal segment $\underline{x}_u$, and looks for possible diagonal segments below it. If a segment below $\underline{x}_b$, $b \neq u$, is found, it looks for a segment $\underline{x}_r$ $r \neq u$, $r \neq b$, on the right from the segment $\underline{x}_b$. In order to qualify as a below segment, we require that $\underline{x}_b(1) > \underline{x}_u(3)$, and that there must be some overlap between the column indices of $\underline{x}_u$ and $\underline{x}_b$. To qualify as a right segment $\underline{x}_r$, there must be some overlap between the row indices of segments $\underline{x}_b$ and $\underline{x}_r$. The groups of three segments fulfilling the above criteria are denoted with $\underline{m}_z = [u, b, r]$, $z = 1,...,Z$. In theory there could be at maximum of $P(P-1)(P-2)$ such groups of three segments, in practice the number is much less. An arbitrary segment may belong to zero or several groups.

The groups of three stripes are then scored based on how close to ideal the group of three stripes is. This scoring affects the scores of some of the segments belonging to these groups. Four partial scores are calculated to measure the quality of each group of three stripes $\underline{m}_z = [u, b, r]$. The first partial score measures how close is the end point of the above segment $\underline{x}_u$ and below segment $\underline{x}_b$:

$$\sigma_1(z) = 1 - 2\left|\underline{x}_u(4) - \underline{x}_b(4)\right| / (\Delta(\underline{x}_b) + \Delta(\underline{x}_u)), \quad (5)$$

where $\underline{x}_u(4)$ and $\underline{x}_b(4)$ are the column indices of the end points of upper and below segments, respectively. The second partial score depends on the vertical alignment of upper and below segments:

$$\sigma_2(z) = \begin{cases} 1-(\underline{x}_u(2)-\underline{x}_b(2))/\Delta(\underline{x}_b) & \text{if } \underline{x}_b(2) < \underline{x}_u(2) \\ 1-(\underline{x}_b(2)-\underline{x}_u(4))/\Delta(\underline{x}_b) & \text{if } \underline{x}_b(2) > \underline{x}_u(4) \\ 1 & \text{otherwise} \end{cases} \quad (6)$$

The next score measures whether the segments $\underline{x}_b$ and $\underline{x}_r$ are of equal length:

$$\sigma_3(z) = 1 - \left|\Delta(\underline{x}_r) - \Delta(\underline{x}_b)\right| / \Delta(\underline{x}_b) . \quad (7)$$

The final partial score depends on the difference in the position of left and right segments:

$$\sigma_4(z) = 1 - \frac{2 \cdot \min(\left|\underline{x}_b(1) - \underline{x}_r(1)\right|, \left|\underline{x}_b(3) - \underline{x}_r(3)\right|)}{\Delta(\underline{x}_b) + \Delta(\underline{x}_r)}, \quad (8)$$

where 'min' denotes minimum operator.

The final score for the group of three segments $\underline{m}_z = [u, b, r]$ is the average of $\sigma_1(z)$, $\sigma_2(z)$, $\sigma_3(z)$, and $\sigma_4(z)$ denoted $\hat{\sigma}(z)$. Since this score considers a segment group, we need to decide whether all the segments in the group receive a score, or whether only certain segments. It was found beneficial to give the score to segment $\underline{x}_b$. The score could also be given to segment $\underline{x}_u$ as it may also correspond to the first instance of the chorus. However, the diagonal stripe corresponding to $\underline{x}_u$ is often longer than the actual chorus, it often consist e.g. of the repeating verse and chorus. It was observed that it gives better results to score the segment $\underline{x}_b$ as its length often more closely corresponds to the correct chorus length. Thus, depending on whether each found segment belong to at least one group of three segments, it will receive a score $s_3(\underline{x}_p) = \max \hat{\sigma}(y)$, $\{y \mid \underline{m}_y(2) = p\}$. The maximum is taken as each segment may belong to more than one group. If a segment $\underline{x}_p$ does not belong to any group of three segments, $s_3(\underline{x}_p) = 0$.

2.8.3. Average energy and distance of a segment

The next criterion $s_4(\underline{x}_p)$ is the average logarithmic energy of the portion of the music signal defined by the column indices of segment $\underline{x}_p$ normalized with the average energy over the whole signal. Using the energy as one criterion gives more weight to such segments that have high average energy, which is often a characteristic of chorus sections. The partial score $s_5(\underline{x}_p)$ takes into account the average distance value during the segment: the smaller the distance during the whole segment the more likely it is that the segment corresponds to a chorus:

$$s_5(\underline{x}_p) = 1 - \phi(\underline{x}_p)/\Phi , \quad (9)$$

where $\phi(\underline{x}_p)$ is the median distance value of the diagonal segment $\underline{x}_p$ in the distance matrix, and Φ is the average distance value over the whole distance matrix.

2.8.4. Number of times the repetition occurs

The last partial score $s_6(\underline{x}_p)$ considers the number of times the repetition occurs. Other diagonal segments locating on top of or below segment $\underline{x}_p$ are indications that the segment defined by the column indices of $\underline{x}_p$ is repeating more than once. To get a score for this, a search is done for all segment candidates $\underline{x}_p$, and a count is made of all those other segments $\underline{x}_q$ which fulfill the condition $\left|\underline{x}_p(2) - \underline{x}_q(2)\right| \le 0.2 \cdot \Delta(\underline{x}_q)$ and $\left|\underline{x}_p(4) - \underline{x}_q(4)\right| \le 0.2 \cdot \Delta(\underline{x}_q)$. The count of other segments $\underline{x}_q$ fulfilling the above criterion is stored as the score for all segment candidates $\underline{x}_p$. When these counts for all segment candidates have been obtained, the values are normalized by dividing with the maximum count, giving the final values for a score $s_6(\underline{x}_p)$ for each segment.

2.8.5. Selecting the most likely chorus segment

The remaining task is to select the most likely chorus segment based on the various criteria. For each segment $\underline{x}_p$, a score is given as

$$\begin{aligned} S(\underline{x}_p) &= 0.5 \cdot s_1(\underline{x}_p) + 0.5 \cdot s_2(\underline{x}_p) \\ &+ s_3(\underline{x}_p) + 0.5 \cdot s_4(\underline{x}_p) + s_5(\underline{x}_p) + 0.5 \cdot s_6(\underline{x}_p). \end{aligned} \quad (10)$$

There is a possibility to optimize the weights in Eq. 10, which we did not fully explore in the fear of over fitting data but just manually selected weights that gave good performance on a small set of music files. The segment $\underline{x}_p$ maximizing the score *S* is selected as the most likely chorus segment. If at least one group of three diagonal stripes fulfilling the criteria of section 2.8.2 has been found, the choice is made among such segments $\underline{x}_o$ for which $s_3(\underline{x}_o) \neq 0$, i.e. those that have been at an appropriate position in at least one group of three diagonal stripes. If no sets of three stripes is found, the selection is made among all the segments by maximizing *S*. In this case the group score $s_3(\underline{x}_p) = 0$ for all segment candidates. The result of this step is an initial chorus segment $\underline{x}_c$.

2.9. Finding the exact location of the chorus

Next the exact location and length of the chorus section is refined using filtering in two or one dimensions. 2D kernels have earlier been used by Shiu et al. to analyze local similarity of the signal by detecting repeated chord successions from a measure-level self-similarity matrix [15]. Here, we use 2D filters to get the exact position for a chorus segment. Often, the time signature in western pop and rock music has a 4/4 time signature, and the length of a chorus section is 8 or 16 measures (32 or 64 beats, respectively) [9]. In addition, the chorus may consist of two repeating subsections of equal length. Two dimensional filter kernels are constructed to model the pattern of ideal small-distance stripes that would be caused by a chorus of 8 or 16 measures long, with two repeating subsections. Figure 6 shows the filter of dimension 32 by 32 beats, with two 16 by 16 beats long repeating subsections. This is the idealized shape of the small-distance stripes occurring in the distance matrix if the song has this kind of chorus. The second filter is

similar but of dimension 64 by 64, and with diagonals modeling the 32 beat long subsections.

The area of the distance matrix surrounding the chorus candidate is filtered with these two kernels. The chorus candidate start column is denoted with $\underline{x}_c(2)$ and the end column $\underline{x}_c(4)$. The columns of the low triangular distance matrix starting from $\max(1, \underline{x}_c(2) - N_f/2)$ to $\min(\underline{x}_c(4) + N_f/2, M)$ are selected as the area from which to search for the chorus. N_f is the dimension of the filter kernel, either 32 or 64, and M is the length of the song in beats. min and max are applied to prevent over indexing. If the length of the area above in the column dimension is shorter than the filter dimension, this step is omitted. The area is limited to lessen the computational load and to prevent the refined chorus section from departing too much from the initial chorus candidate.

When the upper left-hand side corner of the filter with dimension N_f is positioned in (i, j) at the distance matrix, the following values are calculated: mean distance $\alpha(i, j, N_f)$ along the diagonals (marked with black color in Figure 6), mean distance $\beta(i, j, N_f)$ along the main diagonal and mean distance $\lambda(i, j, N_f)$ of the surrounding area (white color in Figure 6). The ratio $\rho_\alpha(i, j, N_f) = \alpha(i, j, N_f) / \lambda(i, j, N_f)$ indicates how well the position matches with a chorus with two identical repeating subsections, and the ratio $\rho_\beta(i, j, N_f) = \beta(i, j, N_f) / \lambda(i, j, N_f)$ how well the position matches a strong repeating section of length N_f with no subsections. The smaller the ratio, the smaller the values on the diagonal compared to the surrounding area. The smallest value of $\rho_\alpha(i, j, N_f)$ and $\rho_\beta(i, j, N_f)$ and the corresponding indices are stored for both filters, i.e. with N_f=32 and N_f =64. These smallest values are denoted by $\rho'_\alpha(N_f)$ and $\rho'_\beta(N_f)$.

Several heuristics are then used to select the final chorus position and length based on the filtering results, or if the conditions are not met then another filtering in one dimension along the initial chorus segment is performed. The final chorus section is selected according to the two dimensional filtering, if the smallest ratios are small enough. The following heuristics are used, although many other alternatives would be possible. These rules below have been obtained via trial and error by experimenting with a subset of 50 songs from our music collection.

If $\rho'_\alpha(64) < \rho'_\alpha(32)$, it indicates a good match with the 64 beat long chorus with two 32 beat long subsections. The chorus starting point is selected according to the column index of the point which minimized $\rho'_\alpha(64)$, and its length is taken as 64 beats. Else, if the length of the initial chorus section is less than 32, the chorus section is adjusted according to the point minimizing $\rho'_\alpha(32)$ only if the chorus beginning would change at maximum one beat from the initial location. Finally, if the length of the initial chorus section is closer to 48 than 32 or 64 and $\rho'_\alpha(32) < \rho'_\alpha(64)$ and $\rho'_\beta(32) < \rho'_\beta(64)$ and the column index of the point minimizing $\rho'_\alpha(32)$ is the same as the point minimizing $\rho'_\beta(32)$, the chorus is

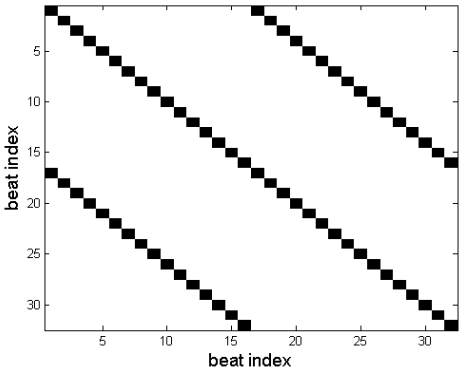


Figure 6. *Two dimensional filter kernel modelling the stripes occurring if the song has a chorus of 32 beats in length with two 16 beat repeating subsections. The average distance is calculated along the entries marked with black colour, and compared to the average distance of locations corresponding to rest of the kernel (white entries).*

set to start at the point minimizing both $\rho'_\alpha(32)$ and $\rho'_\beta(32)$ and its length is set to 32. Thus, these are special rules to adjust the chorus section if it seems likely that there song has either a 32 or 64 beats long chorus with identical subsections half its size.

In many cases, the above conditions are not met, and the chorus section is adjusted by performing filtering along the diagonal values of the initial chorus section and a small offset of five beats before and after its beginning and end. Thus, if the row and column indices of the initial chorus section are denoted with $(\underline{x}_c(1), \underline{x}_c(2))$ (the beginning) and $(\underline{x}_c(3), \underline{x}_c(4))$ (the end), the values to be filtered are found along the line from $(\underline{x}_c(1) - 5, \underline{x}_c(2) - 5)$ to $(\underline{x}_c(3) + 5, \underline{x}_c(4) + 5)$.

The filtering is done with two kernels of length 32 and 64, but now on one dimension along the diagonal distance values of the initial chorus section and its immediate surroundings. The ratio $r(32)$ is the smallest ratio of mean of distance values on the 32 point kernel to the values outside the kernel. If $r(32) < 0.7$ and the length of the initial chorus section is closer to 32 than 64, the chorus starting point is set according to the location minimizing $r(32)$ and its length is set to 32. If the length of the initial chorus section is larger than 48, the final chorus start location and length is selected according to the one giving the smaller score. This step in our method looks for the best position of the chorus section e.g. in the case the diagonal stripe selected as the chorus section consists of a longer repetition of a verse and chorus, for example. Note that the method is not limited to 4/4 time signature and chorus lengths of 32 or 64: if the conditions above are not met, the chorus section is kept as the one returned from the binarization process. In these cases its length does not have to be 32 or 64.

3. EVALUATION

The method was evaluated on database consisting of 206 popular and rock music pieces. Most of the pieces have a clear verse-chorus structure, although there are some instances where the structure is

less obvious. The chorus sections were annotated manually from the pieces. The annotations were made with a dedicated tool, which showed the beat synchronized SDM of the signal aligned with the signal itself. The self-distance matrix visualization significantly speeded up the annotation work as the different sections were more easily found.

Performance of the system is measured with the F-measure, defined as the harmonic mean of the recall rate (R) and precision rate (P): $F = (2RP) / (R + P)$. To calculate R and P, we find the annotated chorus section with maximum overlap with the detected chorus section, and calculate the length l_{corr} of the section where the detected chorus section overlaps with the annotated section. R is calculated as the ratio l_{corr} to the length of the annotated chorus section, and P is the ratio of l_{corr} to the length of the detected chorus section. The F-measure is calculated for each track, and the reported overall F-measure is the average of the F-measures over all tracks.

Table 1 shows the chorus detection results. Baseline is the normal system. The most common error is small offsets in the beginning and/or end locations of the chorus section that reduce the score. The second row represents the results when the output chorus section length is fixed to 30 seconds. Being able to output a fixed length segment may be desirable in some applications, such as music preview. If the initial chorus section is shorter than 30 seconds, expanding is done by following the diagonal chorus segment into the direction of minimum distance in the SDM. Correspondingly, shortening is done by dropping in turn the point with larger distance value from either end. As the recall rate increases when the 30 s limit is applied, the method has not always captured the whole chorus section. If it is desirable that the thumbnail section captures the chorus and it's acceptable if the section extends beyond the chorus, the 30s option can be used. The method is efficient; it takes about ten seconds to process a song with an average duration of three to four minutes on a Windows XP computer with a 2.8 GHz Intel Xeon processor.

Method	P	R	F
Baseline	89%	83%	86%
30s length	70%	92%	79%

Table 1: *Chorus detection results.*

4. CONCLUSIONS

A method for chorus detection from popular and rock music was presented. The method utilizes a novel feature analysis front-end where the MFCC and chroma distance matrices are summed and a two step procedure of initial chorus selection and section refinement. A novel heuristic scoring scheme was proposed to select the initial chorus candidate from the binarized distance matrix, and a novel approach utilizing image processing filters is used to refine the final position and length of the chorus candidate. Evaluations on a manually annotated database of 206 songs demonstrate that the performance of the method is sufficient for practical applications, such as previewing playlists of popular and rock music. Moreover, the method is computationally efficient.

5. REFERENCES

[1] G. Peeters, A. La Burthe, X. Rodet, "Toward Automatic Music Audio Summary Generation from Signal Analysis", in *Proc. of the 3rd International Conference on Music Information Retrieval, ISMIR 2002*, Paris (France), October 2002.

[2] B. Logan, S. Chu, "Music summarization using key phrases," in *Proc. of the IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, *ICASSP 2000*, vol. 2, pp. 749-752, Istanbul, Turkey, May 2000.

[3] M. Levy, M. Sandler, M. Casey, "Extraction of High-Level Musical Structure From Audio Data and Its Application to Thumbnail Generation," in *Proc. IEEE ICASSP 2006*, vol. V, pp. 13-16.

[4] C. Rhodes, Casey, S. Abdallah, M. Sandler, "A Markov-chain monte-carlo approach to musical audio segmentation," in *Proc. IEEE ICASSP 2006*, vol. V, pp. 797-800.

[5] J. Wellhausen and H. Crysandt, "Temporal Audio Segmentation Using MPEG-7 Descriptors," in *Proc. of the SPIE International Symposium on ITCom 2003 - Internet Multimedia Management Systems IV,* Orlando (FL), USA, September 2003.

[6] M. Cooper, J. Foote, "Summarizing Popular Music Via Structural Similarity Analysis," in *Proc. of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, WASPAA 2003*, October 19-22, 2003, New Paltz, NY.

[7] M. A. Bartsch, G. H. Wakefield, "Audio Thumbnailing of Popular Music Using Chroma-Based Representation," *IEEE Trans. on Multimedia*, vol. 7, no. 1, Feb. 2005, pp. 96-104.

[8] M. Goto: "A Chorus Section Detection Method for Musical Audio Signals and Its Application to a Music Listening Station," *IEEE Trans. on Audio, Speech, and Language Processing*, vol. 14, no. 5, Sept. 2006 pp. 1783 – 1794.

[9] N. Maddage, "Automatic Structure Detection for Popular Music," *IEEE Multimedia*, Jan.-March 2006, vol. 13, no. 1, pp. 65-77.

[10] J. Paulus, A. Klapuri, "Music Structure Analysis by Finding Repeated Parts", in *Proc. of the 1st Audio and Music Computing for Multimedia Workshop (AMCMM2006)*, Santa Barbara, California, USA, October 27, 2006, pp. 59-68.

[11] J. Seppänen, A. Eronen, and J. Hiipakka, "Joint Beat & Tatum Tracking from Music Signals", In *Proc. of the 7th International Conference on Music Information Retrieval, ISMIR 2006,* Victoria, Canada, 8 - 12 October 2006.

[12] D. Ellis, "Beat Tracking with Dynamic Programming", MIREX 2006 Audio Beat Tracking Contest system description, Sep 2006, available at http://www.ee.columbia.edu/~dpwe/pubs/Ellis06-beattrack.pdf

[13] M. Marolt, A Mid-level Melody-based Representation for Calculating Audio Similarity, In *Proc. of the 7th International Conference on Music Information Retrieval, ISMIR 2006,* Victoria, Canada, 8 - 12 October 2006.

[14] N. Otsu, A threshold selection method from gray-level histograms, *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-9, no. 1, pp. 62-66, Jan. 1979.

[15] Y. Shiu, H. Jeong, C.-C. Jay Kuo, "Similarity Matrix Processing for Music Structure Analysis", In *Proc. of the 1st Audio and Music Computing for Multimedia Workshop (AMCMM2006)*, October 27, 2006, Santa Barbara, California, USA.

A MATLAB TOOLBOX FOR MUSICAL FEATURE EXTRACTION FROM AUDIO

Olivier Lartillot, Petri Toiviainen

University of Jyväskylä
Finland
lartillo@campus.jyu.fi

ABSTRACT

We present *MIRtoolbox*, an integrated set of functions written in Matlab, dedicated to the extraction of musical features from audio files. The design is based on a modular framework: the different algorithms are decomposed into stages, formalized using a minimal set of elementary mechanisms, and integrating different variants proposed by alternative approaches – including new strategies we have developed –, that users can select and parametrize.

This paper offers an overview of the set of features, related, among others, to timbre, tonality, rhythm or form, that can be extracted with *MIRtoolbox*. Four particular analyses are provided as examples. The toolbox also includes functions for statistical analysis, segmentation and clustering. Particular attention has been paid to the design of a syntax that offers both simplicity of use and transparent adaptiveness to a multiplicity of possible input types. Each feature extraction method can accept as argument an audio file, or any preliminary result from intermediary stages of the chain of operations. Also the same syntax can be used for analyses of single audio files, batches of files, series of audio segments, multichannel signals, etc. For that purpose, the data and methods of the toolbox are organised in an object-oriented architecture.

1. MOTIVATION AND APPROACH

MIRToolbox is a *Matlab* toolbox dedicated to the extraction of musically-related features from audio recordings. It has been designed in particular with the objective of enabling the computation of a large range of features from databases of audio files, that can be applied to statistical analyses.

Few softwares have been proposed in this area. The most important one, Marsyas [1], provides a general architecture for connecting audio, soundfiles, signal processing blocks and machine learning (see section 5 for more details). One particularity of our own approach relies in the use of the *Matlab* computing environment, which offers good visualisation capabilities and gives access to a large variety of other toolboxes. In particular, the *MIRToolbox* makes use of functions available in recommended public-domain toolboxes such as the *Auditory Toolbox* [2], *NetLab* [3], or *SOM-toolbox* [4]. Other toolboxes, such as the *Statistics toolbox* or the *Neural Network toolbox* from MathWorks, can be directly used for further analyses of the features extracted by *MIRToolbox* without having to export the data from one software to another.

Such computational framework, because of its general objectives, could be useful to the research community in Music Information Retrieval (MIR), but also for educational purposes. For that reason, particular attention has been paid concerning the ease of use of the toolbox. In particular, complex analytic processes can be designed using a very simple syntax, whose expressive power comes from the use of an object-oriented paradigm.

The different musical features extracted from the audio files are highly interdependent: in particular, as can be seen in figure 1, some features are based on the same initial computations. In order to improve the computational efficiency, it is important to avoid redundant computations of these common components. Each of these intermediary components, and the final musical features, are therefore considered as *building blocks* that can been freely articulated one with each other. Besides, in keeping with the objective of optimal ease of use of the toolbox, each building block has been conceived in a way that it can adapt to the type of input data. For instance, the computation of the MFCCs can be based on the waveform of the initial audio signal, or on the intermediary representations such as spectrum, or mel-scale spectrum (see Fig. 1). Similarly, autocorrelation is computed for different range of delays depending on the type of input data (audio waveform, envelope, spectrum). This decomposition of all the set of feature extraction algorithms into a common set of building blocks has the advantage of offering a synthetic overview of the different approaches studied in this domain of research.

2. FEATURE EXTRACTION

2.1. Feature overview

Figure 1 shows an overview of the main features implemented in the toolbox. All the different processes start from the audio signal (on the left) and form a chain of operations proceeding to right. The vertical disposition of the processes indicates an increasing order of complexity of the operations, from simplest computation (top) to more detailed auditory modelling (bottom).

Each musical feature is related to one of the musical dimensions traditionally defined in music theory. Boldface characters highlight features related to pitch, to tonality (chromagram, key strength and key Self-Organising Map, or SOM) and to dynamics (Root Mean Square, or RMS, energy). Bold italics indicate features related to rhythm, namely tempo, pulse clarity and fluctuation. Simple italics highlight a large set of features that can be associated to timbre. Among them, all the operators in grey italics can be in fact applied to many others different representations: for instance, statistical moments such as centroid, kurtosis, etc., can be applied to either spectra, envelopes, but also to histograms based on any given feature.

One of the simplest features, zero-crossing rate, is based on a simple description of the audio waveform itself: it counts the number of sign changes of the waveform. Signal energy is computed using root mean square, or RMS [5]. The envelope of the audio signal offers timbral characteristics of isolated sonic event.

FFT-based spectrum can be computed along the frequency domain or along Mel-bands, with linear or decibel energy scale, and

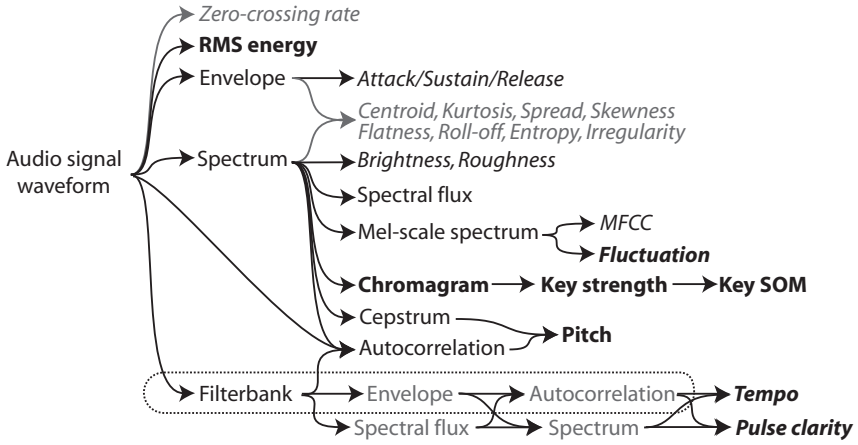


Figure 1: *Overview of the musical features that can be extracted with* MIRToolbox.

applying various windowing methods. The results can be multiplied with diverse resonance curves in order to highlight different aspects such as metrical pulsation (when computing the FFT of envelopes) or fluctuation [6].

Many features can be derived from the FFT:

- Basic statistics of the spectrum gives some timbral characteristics (such as spectral centroid, roll-off [5], brightness, flatness, etc.).
- The temporal derivative of spectrum gives the spectral flux.
- An estimation of roughness, or sensory dissonance, can be assessed by adding the beating provoked by each couple of energy peaks in the spectrum [7].
- A conversion of the spectrum in a Mel-scale can lead to the computation of Mel-Frequency Cepstral Coefficients (MFCC) (cf. example 2.2), and of fluctuation [6].
- Tonality can also be estimated (cf. example 2.3).

The computation of the autocorrelation can use diverses normalization strategies, and integrates the improvement proposed by Boersma [8] in order to compensate the side-effects due to the windowing. Resonance curve are also available here. Autocorrelation can be "generalized" through a compression of the spectral representation [9].

The estimation of pitch is usually based on spectrum, autocorrelation, or cepstrum, or a mixture of these strategies [10].

A distinct approach consists of designing a complete chain of processes based on the modelling of auditory perception of sound and music [2] (circled in Figure 1). This approach can be used in particular for the computation of rhythmic pulsation (cf. example 2.4).

2.2. Example: Timbre analysis

One common way of describing timbre is based on MFCCs [11, 2]. Figure 2 shows the diagram of operations. First, the audio sequence is loaded (1), decomposed into successive frames (2), which are then converted into the spectral domain, using the mirspectrum function (3). The spectra are converted from the frequency domain to the Mel-scale domain: the frequencies are rearranged into 40 frequency bands called Mel-bands[1]. The envelope of the Mel-scale spectrum is described with the MFCCs, which are obtained by applying the Discrete Cosine Transform to the Mel-scale spectrum. Usually only a restricted number of them (for instance the 13 first ones) are selected (5).

```
a = miraudio('audiofile.wav')      (1)
f = mirframe(a)                    (2)
s = mirspectrum(f)                 (3)
m = mirspectrum(s,'Mel')           (4)
c = mirmfcc(s,'Rank',1:13)         (5)
```

The computation can be carried in a window sliding through the audio signal (this corresponded to the code line 1), resulting in a series of MFCC vectors, one for each successive frame, that can be represented column-wise in a matrix. Figure 2 shows an example of such matrix. The MFCCs do not convey very intuitive meaning per se, but are generally applied to distance computation between frames, and therefore to segmentation tasks (cf. paragrapn 2.5).

The whole process can be executed in one single line by calling directly the mirmfcc function with the audio input as argument:

```
mirmfcc(f,'Rank',1:13)             (6)
```

2.3. Example: Tonality analysis

The spectrum is converted from the frequency domain to the pitch domain by applying a log-frequency transformation. The distribution of the energy along the pitches is called the *chromagram*. The chromagram is then wrapped, by fusing the pitches belonging to same pitch classes. The wrapped chromagram shows therefore a distribution of the energy with respect to the twelve possible pitch classes [12].

Krumhansl and Schmuckler [13] proposed a method for estimating the tonality of a musical piece (or an extract thereof)

[1] The Mel-scale conversion is available as an option of the mirspectrum function (4). Note how it is possible to recall a function using one of its previous output as input (here, s), in order to perform some additional optional operations.

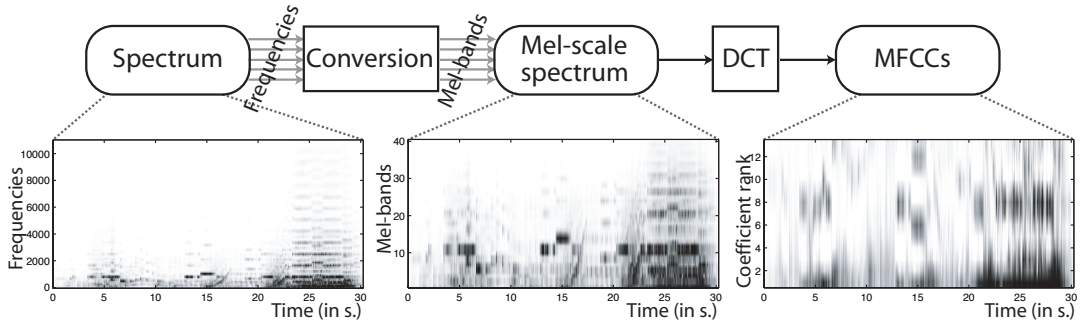


Figure 2: *Successive steps for the computation of MFCCs, illustrated with the analysis of an audio excerpt decomposed into frames.*

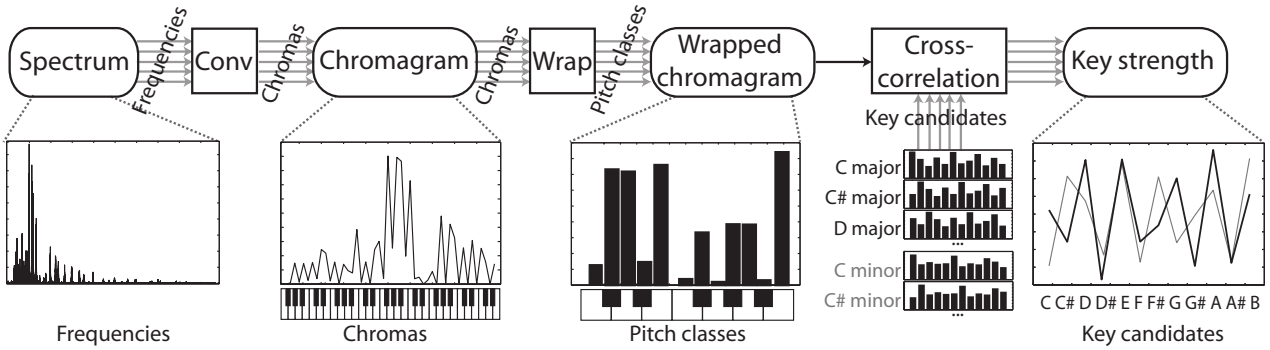


Figure 3: *Successive steps for the calculation of chromagram and estimation of key strengths, illustrated with the analysis of an audio excerpt, this time not decomposed into frames.*

by computing the cross-correlation of its pitch class distribution with the distribution associated to each possible tonality. These distribution have been established though listening experiments [14]. The most prevalent tonality is considered to be the tonality candidate with highest correlation, or *key strength*. This method was originally designed for the analysis of symbolic representations of music but has been extended to audio analysis through an adaptation of the pitch class distribution to the chromagram representation [12]. Figure 3 displays the successive steps of this approach. For instance the following command estimates the three most probable key candidates for each frame.

```
mirkey(f,'Total',3)
```
(7)

A richer representation of the tonality estimation can be drawn with the help of a self-organizing map (SOM), trained by the 24 tonal profiles [15]. The configuration of the trained SOM reveals key relations that correspond to music theoretical notions. The estimation of the tonality of the musical piece under study is carried by projecting its wrapped chromagram onto the SOM. Figure 4 shows the resulting activity pattern in the SOM.

2.4. Example: Rhythm analysis

One common way of estimating the rhythmic pulsation, described in figure 6, is based on auditory modelling [5]. The audio signal is first decomposed into auditory channels using a bank of filters. Diverse types of filterbanks are proposed and the number of channels can be changed, such as 20 for instance (8). The envelope of each

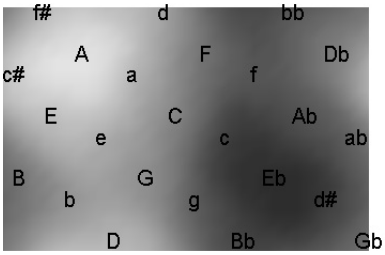


Figure 4: *Activity pattern of a self-organizing map representing the tonal configuration of the first two seconds of Mozart* Sonata in A major*, K 331. High activity is represented by bright nuances.*

channel is extracted (9)[2]. As pulsation is generally related to increase of energy only, the envelopes are differentiated, half-wave rectified, before being finally summed together again (10). This gives a precise description of the variation of energy produced by each note event from the different auditory channels.

After this onset detection, the periodicity is estimated through autocorrelation (12)[3]. However, if the tempo varies throughout the piece, an autocorrelation of the whole sequence will not show clear periodicities. In such cases it is better to compute the autocorrela-

[2]Note how the analysis of multi-channel signal (such as fb) follows exactly the same kind of syntax than for mono-channel signal.

[3]For the sake of clarity, several options in the following functions have been omitted.

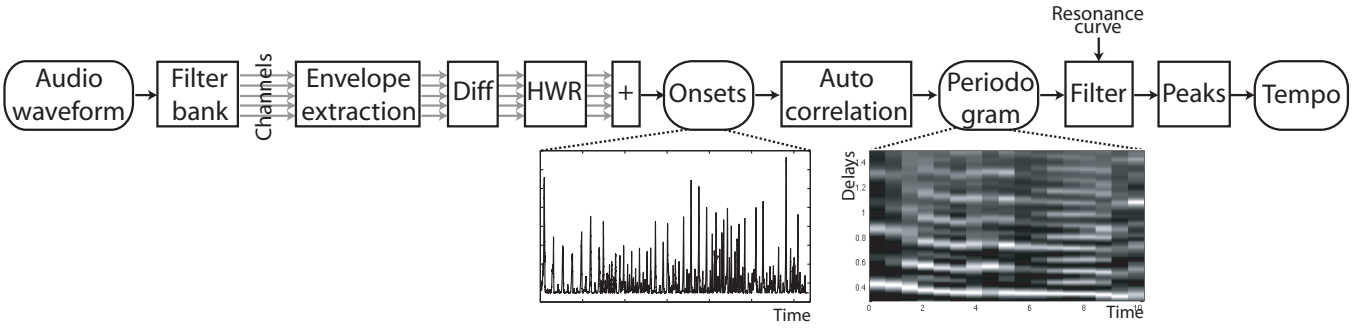


Figure 5: *Successive steps for the estimation of tempo illustrated with the analysis of an audio excerpt. In the periodogram, high autocorrelation values are represented by bright nuances.*

tion for a frame decomposition (11)[4]. This yields a periodogram that highlights the different periodicities, as shown in figure 6. In order to focus on the periodicities that are more perceptible, the periodogram is filtered using a resonance curve [16] (12), after which the best tempos are estimated through peak picking (13), and the results are converted into beat per minutes (14). Due to the difficulty of choosing among the possible multiples of the tempo, several candidates (three for instance) may be selected for each frame, and a histogram of all the candidates for all the frames, called periodicity histogram, can be drawn (15).

```
fb = mirfilterbank(a,20)                    (8)
e = mirenvelope(fb,'Diff','Halfwave')       (9)
s = mirsum(e)                               (10)
fr = mirframe(s,3,.1)                       (11)
ac = mirautocor(fr,'Resonance')             (12)
p = mirpeaks(ac,'Total',1,'NoEnd')          (13)
t = mirtempo(p)                             (14)
h = mirhisto(t)                             (15)
```

The whole process can be executed in one single line by calling directly the mirtempo function with the audio input as argument:

```
mirtempo(a,'Frame')                         (16)
```

In this case, the different options available throughout the process can directly be specified as argument of the tempo function. For instance, a computation of a frame-based tempo estimation, with a selection of the 3 best tempo candidates in each frame, a range of admissible tempi between 60 and 120 beats per minute, an estimation strategy based on a mixture of spectrum and autocorrelation applied on the spectral flux will be executed with the syntax:

```
mirtempo(a,'Frame','Total',3,
         'Min',60,'Max',120,'Spectrum',
         'Autocor','SpectralFlux')          (17)
```

[4]The mirframe function can accept both audio signal and envelope as argument. Here, the frame size is 3 seconds and the hop factor .1.

2.5. Segmentation

More elaborate tools have also been implemented that can carry out higher-level analyses and transformations. In particular, audio files can be automatically segmented into a series of homogeneous sections, through the estimation of temporal discontinuities along diverse alternative features such as timbre in particular [17]. First the audio signal is decomposed into frames (18) and one chosen feature, such as MFCC (19), is computed along these frames. The feature-based distances between all possible frame pairs are stored in a similarity matrix (20). Convolution along the main diagonal of the similarity matrix using a Gaussian checkerboard kernel yields a novelty curve that indicates the temporal locations of significant textural changes (21). Peak detection applied to the novelty curve returns the temporal position of feature discontinuities (22) that can be used for the actual segmentation of the audio sequence (23)[5].

```
fr = mirframe(a)                            (18)
fe = mirmfcc(fr)                            (19)
sm = mirsimatrix(fe)                        (20)
nv = mirnovelty(sm)                         (21)
ps = mirpeaks(nv)                           (22)
sg = mirsegment(a,ps)                       (23)
                                            (24)
```

The whole segmentation process can be executed in one single line by calling directly the mirsegment function with the audio input as argument:

```
mirsegment(a,'Novelty')                     (25)
```

By default, the novelty curve is based on MFCC, but other features can be selected as well using an additional option:

```
mirsegment(a,'Novelty','Spectrum')          (26)
```

A second similarity matrix can be computed, in order to show the distance – according to the same feature than the one used for

[5]The first argument of the mirsegment function is the audio file that needs to be segmented. It is possible for instance to compute the novelty curve using a downsampled version of `a` (18) and to perform the actual segmentation using the original audio file.

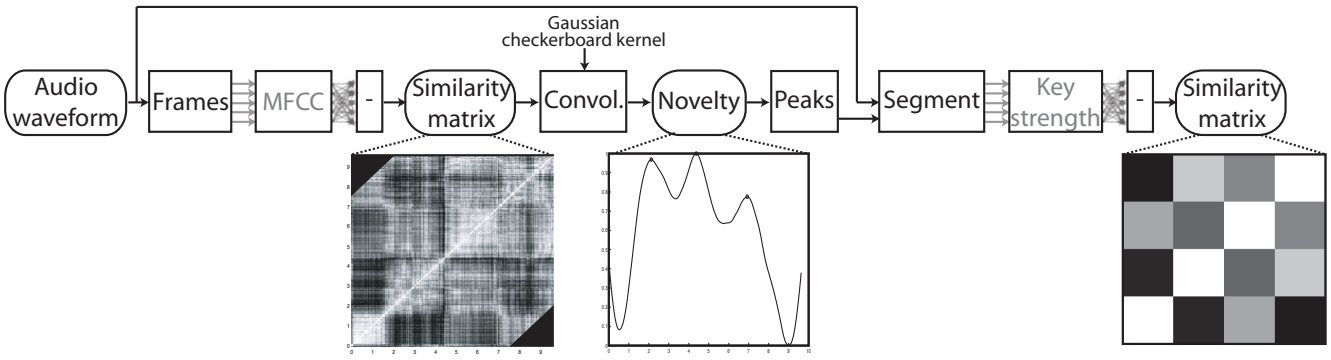


Figure 6: *Successive steps for the segmentation of an audio sequence based on timbral novelty. In the similarity matrix, high similarity values are represented by bright nuances.*

the segmentation – between all possible segment pairs (28).[6]

```
fesg = mirmfcc(sg)          (27)
smsg = mirsimatrix(fesg)    (28)
```

2.6. Data analysis

The toolbox includes diverse tools for data analysis, such as a peak extractor, and functions that compute histograms, entropy, zero-crossing rates, irregularity or various statistical moments (centroid, spread, skewness, kurtosis, flatness) on data of various types, such as spectrum, envelope or histogram.

The `mirpeaks` functions can accept any data returned by any other function of the MIRtoolbox and can adapt to the different kind of data of any number of dimensions. In the graphical representation of the results, the peaks are automatically located on the corresponding curves (for 1D data) or bit-map images (for 2D data).

The `mirpeaks` functions offers alternative possible heuristics. It is possible to define a global threshold that peaks must exceed for them to be selected. We have designed a new strategy of peak selection, based on a notion of *contrast*, discarding peaks that are not sufficiently contrastive (based on a certain threshold) with the neighbouring peaks. This adaptive filtering strategy hence adapts to the local particularities of the curves. Its articulation with other more conventional thresholding strategies leads to an efficient peak picking module that can be applied throughout the *MIRtoolbox*.

Supervised classification of musical samples can also be performed, using techniques such as K-Nearest Neighbours or Gaussian Mixture Model. One possible application is the classification of audio recordings into musical genres.

3. DESIGN OF THE TOOLBOX

3.1. Data encapsulation

All the data returned by the functions in the toolbox are encapsulated into types objects. The default display method associated to all these objects is a graphical display of the corresponding curves. In this way, when the display of the values of a given analysis is requested, what is printed is not a listing of long vectors or matrices, but rather a correctly formatted graphical representation.

The actual data matrices associated to those data can be obtained by calling a method called `mirgetdata`, which constructs the simplest possible data structure associated to the data (cf. paragraph 4.1).

3.2. Frame analysis

Frame-based analyses (i.e., based on the use of a sliding window) can be specified using two alternative methods. The first method is based on the use of the `mirframe` function, which decomposes an audio signal into successive frames. Optional arguments can specify the frame size (in seconds, by default), and the hop factor (between 0 and 1, by default). For instance, in the following code (line 29), the frames have a size of 50 milliseconds and are half overlapped. The results of that function could then be directly sent as input of any other function of the toolbox (30):

```
f = mirframe(a,.05,.5)      (29)
mirtempo(f)                 (30)
```

Yet this first method does not work correctly for instance when dealing with tempo estimation as described in section 2.4. Following this first method, as shown in figure 7, the frame decomposition is the first step performed in the chain of processes. As a result, the input of the filterbank decomposition is a series of short frames, which induces two main difficulties. Firstly, in order to avoid the presence of undesirable transitory state at the beginning of each filtered frame, the initial state of each filter would need to be tuned depending on the state of the filter at one particular instant of the previous frame (depending of the overlapping factor). Secondly, the demultiplication of the redundancies of the frame decomposition (if the frames are overlapped) throughout the multiple channels of the filterbank would require the use of consequent memory space. The technical difficulties and waste of memory induced by this first method can be immediately overcome if the frame decomposition is performed after the filterbank decomposition and recomposition, as shown in figure 8.

This second method, more successful in this context, cannot be managed using the previous syntax, as the input of the `mirtempo` function should not be frame-decomposed yet. The other alternative syntax consists in proposing the frame decomposition option

[6] Note how the computation of a feature along the successive segments of an audio sequence (27) follows exactly the same kind of syntax that for the computation of a feature along successive frames (19).

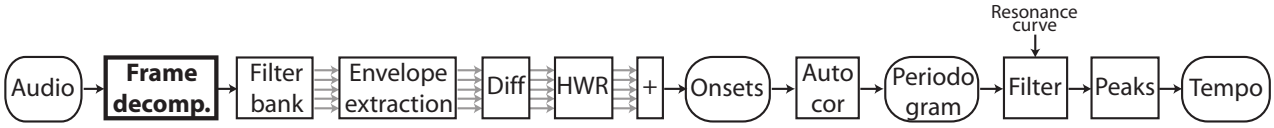


Figure 7: *First possible location of the frame decomposition step (in bold) within the chain of processes defining the tempo estimation method.*

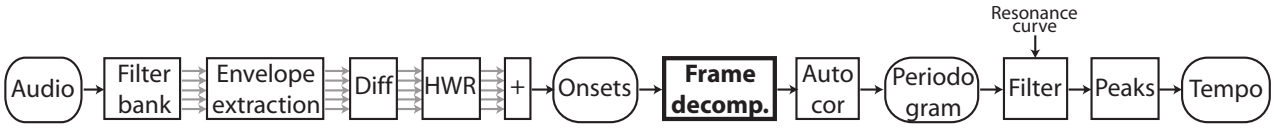


Figure 8: *Second – and more suitable – possible location of the frame decomposition step (in bold), once the onset detection curve has been computed.*

as a possible argument (`'Frame'`) of the `mirtempo` function (31). This corresponds to what was presented in section 2.4 (code lines 16 and 17).

```
mirtempo(a,'Frame',.05,.5)          (31)
```

The frame decomposition option is available as a possible argument to most of the functions of the toolbox. Each function can then specify the exact position of the frame decomposition within its chain of operations. Besides, if not specified, the default parameters of the frame decomposition – i.e., frame size and hop factor – can be adapted to each specific function. Hence, from a user's point of view, the execution and chaining of the different operators of the MIRtoolbox follow the same syntax, be there frame decomposition or not, apart from the additional use of either the command `mirframe` or the option `'Frame'` for frame decomposition. Of course, from a developer's point of view, this requires that each feature extraction algorithm should adapt to frame-decomposed input. More precisely, as will be explained in section 4.1, input can be either a single vector or a matrix, where columns represent the successive frames. Conveniently enough, in the Matlab environment, the generalization of vector-based algorithms to matrix-based versions is generally effortless.

3.3. Adaptive syntax

As explained previously, the diverse functions of the toolbox can accept alternative input:

- The name of a particular audio file (either in wav or au format) can be directly specified as input:

```
mirspectrum('myfile')          (32)
```

- The audio file can be first loaded using the `miraudio` function, which can perform diverse operations such as resampling, automated trimming of the silence at the beginning and/or at the end of the sequence, extraction of a given subsequence, centering, normalization with respect to RMS energy, etc.

```
a = mirtempo('myfile','Sampling',11025,
             'Trim','Extract',2,3,
             'Center','Normal')        (33)
mirspectrum(a)                         (34)
```

- Batch analyses of audio files can be carried out by simply replacing the name of the audio file by the keyword `'Folder'`.

```
mirspectrum('Folder')          (35)
```

- Any vector `v` computed in Matlab can be converted into a waveform using, once again, the `miraudio` function, by specifying a specific sampling rate.

```
a = miraudio{v,44100)          (36)
mirspectrum(a)                 (37)
```

- Any feature extraction can be based on the result of a previous computation. For instance, the autocorrelation of a spectrum curve can be computed as follows:

```
s = mirspectrum(a)             (38)
as = mirautocor(s)             (39)
```

- Product of curves [10] can be performed easily:

```
mirautocor(a)*mirautocor(s)    (40)
```

  In this particular example, the waveform autocorrelation `mirautocor(a)` is automatically converted to frequency domain in order to be combined with the spectrum autocorrelation `mirautocor(s)`.

4. IMPLEMENTATION DETAILS

4.1. Data representation

All data returned by the toolbox is represented using the same general framework:

- The one-dimensional analysis of a given frame or of a whole signal is stored in a column vector, which corresponds to the first dimension in Matlab convention.
- The multiple columns corresponding to successive frame analyses are arranged row-wise (along the second dimension in Matlab convention), forming a matrix. Respectively, any two-dimensional data (such as a self-organizing map) is stored in a same matrix using the first two Matlab dimensions.
- The multiple matrices corresponding to multiple channels, when applicable (9), are arranged along the third dimension in Matlab convention, forming a 3D-matrix.
- The fourth Matlab dimension is sometimes used for more complex data. For instance, the keystrength function returns two sets of data – one for major keys, one for minor keys - that are arranged following the fourth dimension.
- These matrices (one to four-dimensional) are computed for each successive segments of a segmented audio file, when applicable (27), and stored in a Matlab cell array.
- The multiple cell arrays corresponding to the analyses of the multiple audio files of a batch of audio files are stored in another cell array.

Figure 9 shows the overall structure.

This complex data structure, although enabling to grasp all the potentiality offered by the toolbox, is rarely used in its plain capacity. Therefore, a particular mechanism has been designed in order to automatically simplify the structure, when calling the $mirgetdata$ function that return the numerical data associated to a given feature analysis.

4.2. Object-oriented architecture

The organization of the data and functions of the mirtoolbox is founded on an object-oriented architecture. The superclass from which all the data and methods are based is called `mirdata`. It contains all the information commonly used by all data.A hierarchy of classes is contructed from the *mirdata* hyperclass. The *miraudio* and *mirenvelope* classes inherit from the *mirtemporal* class, which contains particular data and methods adapted to waveforms of diverse sampling rates. For instance, the *mirplay* method plays back the audio signal. When applied to an envelope, *mirplay* actually produces a white noise featuring the same envelope.

A large number of features actually returns a single scalar value per analysed frame. They are all members of the *mirscalar* class, which features all the necessary methods for their processing, such as their graphical display in particular. The non-scalar features, on the contrary, are organized into a set of different specialised classes (*mirautocor*, *mirspectrum*, *mirhisto*, *mirmfcc*, etc.).

4.3. Memory optimization

The flexibility of the syntax requires a complex data representation that can handle alternative configurations (frame and/or channels decompositions, segmentation, batch analysis). This data structure could in theory become very extensive in terms of memory usage, especially if entire folders of audio files are loaded into the memory in one go. We have designed new methods allowing a better management of memory without deterioration of the syntactical simplicity and power. Audio files are loaded one after the other in the memory, and if necessary, long audio files are also divided into a series of successive blocks of frames that are loaded one after the other. We plan to further optimise the computational efficiency of the toolbox by proposing the possibility of distributing the computational loads among a network of computers, with the help of the *Distributed Computing* Toolbox and Engine proposed by Matlab.

4.4. Software Development Kit

The different feature extraction algorithms will be progressively refined and new features will be added in future versions of *MIRtoolbox*. Users are encouraged to write their own functions, using the building blocks offered by the current version. A set of meta-functions have been designed that enable the writing of additional algorithms using very simple function templates. As the meta-functions take care of all the complex management of the data structure and methods, the development of new algorithms can concentrate simply on the purely mathematical and DSP considerations. This may result in a computational environment where large-scale MIR systems could be developed, articulated one with each other, and compared.

5. *MIRTOOLBOX* COMPARISON TO *MARSYAS*

Marsyas is a framework written in C++ and Java for prototyping and experimentation with computer audition applications [1]. It provides a general architecture for connecting audio, soundfiles, signal processing blocks and machine learning. The architecture is based on dataflow programming, where computation is expressed as a network of processing nodes/components connected by a number of communication channels/arcs. Users can build their own dataflow network using a scripting language at run-time. Marsyas provides a framework for building applications rather than a set of applications [1] [7] Marsyas executables operate either on individual soundfiles or collections which are simple text files that contain lists of soundfiles. In general collection files should contain soundfiles with the same sampling rate as Marsyas doesn't perform automatic sampling conversion (except between 44100Hz and 22050Hz). The results of feature extraction processes are stored in Marsyas as text files that can be used later in the Weka machine learning environment. In parallel, Marsyas integrates some basic machine learning components.

Also MIRtoolbox offers the possibility of articulating process one after the other in order to construct complex computation, using a simple and adaptive syntax. Contrary to Marsyas though, MIRtoolbox does not offer real-time capabilities. On the other hand, its object-based architecture (paragraph 4.2) enables a significant simplification of the syntax. MIRtoolbox can also analyse folders of audio files, and can deal with folder of varying sampling rates without having to perform any conversion. The data computed by the MIRtoolbox can be further processed directly in the

[7] The main features currently proposed are spectral moments, flux, and rolloff, pitch and harmonicity estimation, MFCC and LPC, zero-crossing and RMS.

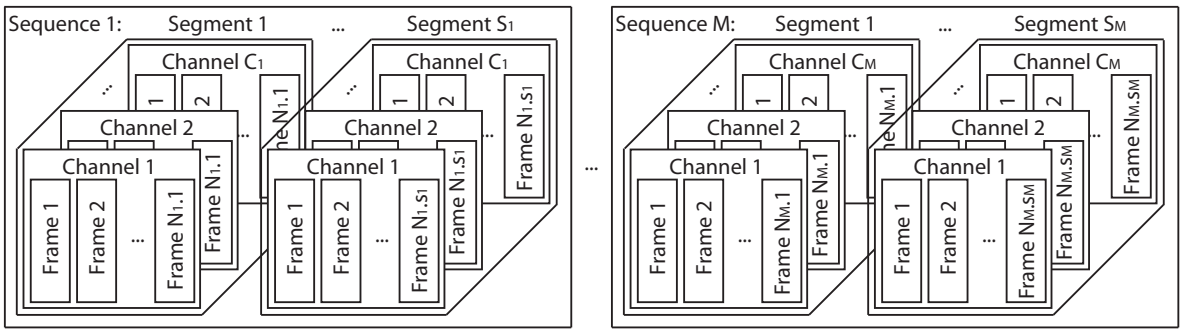


Figure 9: *Structure of the data representation used for each feature results.*

Matlab environment with the help of other toolboxes, or can be exported into text files.

6. AVAILABILITY OF THE *MIRTOOLBOX*

Following our first Matlab toolbox, called *MIDItoolbox* [18], dedicated to the analysis of symbolic representations of music, the *MIRtoolbox* is offered for free to the research community. It can be downloaded from the following URL:

`http://www.cc.jyu.fi/~lartillo/mirtoolbox`

7. ACKNOWLEDGMENTS

This work has been supported by the European Commission (NEST project "Tuning the Brain for Music", code 028570). The development of the toolbox has benefitted from productive collaborations with the other partners of the project, in particular Tuomas Eerola, Jose Fornari, Marco Fabiani, and students of our department.

8. REFERENCES

[1] G. Tzanetakis and P. Cook, "Marsyas: A framework for audio analysis," *Organized Sound*, vol. 4, no. 3, 2000.

[2] M. Slaney, "Auditory toolbox version 2," Tech. Rep., Interval Research Corporation, 1998-010, 1998.

[3] I. Nabney, *Springer Advances In Pattern Recognition Series*, chapter NETLAB: Algorithms for pattern recognition, 2002.

[4] J. Vesanto, "Proceedings of the matlab dsp conference," in *Self-Organizing Map in Matlab: the SOM Toolbox*, 1999, pp. 35–40.

[5] G. Tzanetakis and P. Cook, "Multifeature audio segmentation for browsing and annotation," in *Proceedings of the 1999 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 1999.

[6] A. Rauber E. Pampalk and D. Merkl, "Content-based organization and visualization of music archives," in *Proceedings of the 10th ACM International Conference on Multimedia*, 2002, pp. 570–579.

[7] E. Terhardt, "On the perception of periodic sound fluctuations (roughness)," *Acustica*, vol. 30, no. 4, pp. 201–213, 1974.

[8] P. Boersma, "Accurate short-term analysis of the fundamental frequency and the harmonics-to-noise ratio of a sampled sound," *IFA Proceedings*, vol. 17, pp. 97–110, 1993.

[9] T. Tolonen and M. Karjalainen, "A computationally efficient multipitch analysis model," *IEEE Transactions on Speech and Audio Processing*, vol. 8, no. 6, pp. 708–716, 2000.

[10] G. Peeters, "Music pitch representation by periodicity measures based on combined temporal and spectral representations," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 2006.

[11] L. Rabiner and B. H. Juangl, *Fundamentals of Speech Recognition*, Prentice-Hall, 1993.

[12] E. Gomez, "Tonal description of polyphonic audio for music content processing," *INFORMS Journal on Computing*, vol. 18, no. 3, pp. 294–304, 2006.

[13] C. Krumhansl, *Cognitive Foundations of Musical Pitch*, Oxford University Press, 1990.

[14] C. Krumhansl and E. J. Kessler, "Tracing the dynamic changes in perceived tonal organization in a spatial representation of musical keys," *Psychological Review*, vol. 89, pp. 334–368, 1982.

[15] P. Toiviainen and C. Krumhansl, "Measuring and modeling real-time responses to music: The dynamics of tonality induction," *Perception*, vol. 32, no. 6, pp. 741–766, 2003.

[16] P. Toiviainen and J.S. Snyder, "Tapping to bach: Resonance-based modeling of pulse," *Music Perception*, vol. 21, no. 1, pp. 43–80, 2003.

[17] J. Foote and M. Cooper, "Media segmentation using self-similarity decomposition," in *Proceedings of SPIE Storage and Retrieval for Multimedia Databases*, 2003, number 5021, pp. 167–175.

[18] T. Eerola and P. Toiviainen, "MIR in Matlab: The Midi Toolbox," in *Proceedings of 5th International Conference on Music Information Retrieval*, 2004, pp. 22–27.

[19] P. N. Juslin, "Emotional communication in music performance: A functionalist perspective and some data," *Music Perception*, vol. 14, pp. 383–418, 1997.

[20] K. R. Scherer and J. S. Oshinsky, "Cue utilization in emotion attribution from auditory stimuli," *Motivation and Emotion*, vol. 1, no. 4, pp. 331–346, 1977.

REAL-TIME VISUALISATION OF LOUDNESS ALONG DIFFERENT TIME SCALES

Esben Skovenborg
TC Group Research A/S
Sindalsvej 34, DK-8240 Risskov, Denmark
EsbenS@TCElectronic.com

Søren H. Nielsen
TC Group Research A/S
Sindalsvej 34, DK-8240 Risskov, Denmark
SHN@TCElectronic.com

ABSTRACT

We propose a set of design criteria for visualising loudness features of an audio signal, measured along different time scales. A novel real-time loudness meter, based on these criteria, is presented. The meter simultaneously shows short-term loudness, long-term loudness and peak level. The short-term loudness is displayed using a circular bar graph. The meter displays the long-term loudness by means of a circular envelope graph, organized according to an absolute time-scale – looking similar to a radar display. Typically, the loudness measured during the past hour is visible. The algorithms underlying the meter's loudness and peak level measurements take into account recent ITU-R recommendations and research into loudness modelling.

1. INTRODUCTION

Time-varying *features* of an audio signal can be visualised in different ways. Such features can be objective measures or they may represent perceptual properties of the signal. The features discussed in this paper are one of both kinds: 1) the perceptual feature *loudness*, and 2) the objective measure *true peak value*.

In the measurement of the features, the analysis of the audio signal is done over time such that the features are represented by time-varying scalars (i.e. vectors) – one for each feature. The basic time resolution of the analysis should be adapted to perceptually or technically relevant granularity. This aim might be in conflict with the possibilities of a suitable visualisation. The big challenge of visualisation is to present the desired amount of information in such a way that it is easy to comprehend – and without losing access to details.

1.1. Time scales for display

For our application, simultaneous display along three different time scales is desirable. One time scale is instantaneous value, reacting quickly to the measured feature and typically holding the indication of a possible alarm condition (e.g. overload) for a short while, to allow an operator to see it.

A more slowly moving indication is useful to assist an operator in adjusting the sound system, typically the gain. This indication should react and move with a speed similarly to an overall perception of the feature. For example, speech from a trained speaker may be considered to be of constant loudness even though short-term fluctuations occur. The display should reflect this fact.

Finally, a log or history of the fluctuations of a feature may be desirable. Such a log could, for instance, be used to verify that the loudness is aligned appropriately between different segments of a broadcast.

This paper presents a prototype of a novel real-time loudness meter, simultaneously showing short-term loudness, long-term loudness and peak level. The three metering functions have been chosen to fulfil needs in broadcasting, as well as in other production environments where a diversity of program material needs to be aligned in perceived level while also being kept within technical limits. A meter in itself does not align the levels – an operator is (ideally) present to attend to the adjustments, assisted by visual tools like meters.

In our design of the meter display, the analogue clock and radar displays were used as inspiration for the visualisation of magnitude and time dimensions.

1.2. Standardisation

Within the ITU-R (International Telecommunication Unit – Radio Communications Sector), a study group has been working on the methods of loudness and true peak level metering, and recently come up with two new recommendations: [1] and [2]. The former describes the measurement algorithms, whereas [2] describes the visual presentation of the measurements. The need for a short-term as well as a long-term loudness measure is recognized, but with only the long-term measurement method specified at present. Furthermore, methods to reliably estimate the true peak value are described.

The visualisation paradigm presented in this paper is an alternative – or supplement – to the one described in [2].

2. THE NEED FOR LOUDNESS MEASUREMENT

The audio content in the numerous formats in use, is dynamically, spectrally, and sometimes even spatially processed according to the properties of the media, format, and playback conditions, see e.g. [3, 4]. Each format requires different optimum settings of bandwidth, dynamic range etc., based on the expected listening conditions and also on the properties of the available transmission channel or storage medium such as data rate.

These different optimum settings come in addition to the dynamics processing needed to reduce undesired loudness variations. Therefore, a *loudness* meter is required as a complement to traditional *level* metering.

In many cases, a fully automatic way of setting processing parameters according to the different requirements would be desirable. This goal may not be trivial to achieve, but in all cases a monitoring function is needed: A meter which can display the relevant perceptual properties, i.e. the short-term and the long-term loudness. Furthermore, a function to monitor the measured peak level is required as an aid to avoid clipping. Such a technical

measure is required, in addition to the perceptual measures (of loudness), due to the limitations of the transmission channel.

2.1. Previous level meters: VU and PPM

Traditionally, the primary purpose of level meters has been of technical nature: They serve as an aid in fulfilling certain technical criteria, such as obtaining a good signal-to-noise ratio on an analogue medium. Here, the standardised VU- and PPM-types of level meters are discussed [5, 6, 7].

The VU (volume unit) meter [5] measures the full-wave rectified (i.e., absolute value) level with a relatively slow time constant. The response time of the meter to rising and falling levels is (ideally) identical. For judging the overall level the VU meter can be quite useful, but due to its measurement algorithm the VU meter is not suitable for loudness measurement. With the soft saturation characteristics of analogue tape recordings in mind, the VU meter has been successfully used for years to set the right recording level – often supplemented by a peak-indicating lamp, as the meter is too slow to react on short transients. The scale of the VU meter is shown in Figure 1. A mechanical instrument with a thin indicator needle is typically employed. Note the contrast in the meter, and the curved scale with approximately linear voltage scale and thus non-linear intervals on the dB scale. The overload section (above 100%) is coloured red.

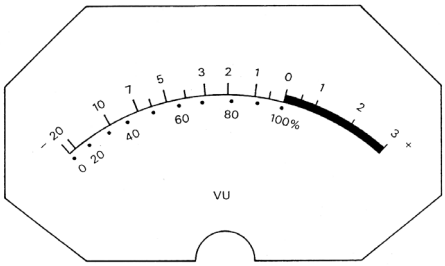


Figure 1: *VU meter scale from [5].*

For more precise control of peak levels, such as needed in radio and TV broadcast for technical and legal reasons, another type of meter was created: the peak programme meter (PPM) [6], [7]. Actually, two generations of PPMs exist: One with instantaneous response to rising levels and another with a short response time (a few milliseconds). The measurement algorithm consists of taking the peak value of the full-wave rectified signal. The decay time is chosen to be long enough that an operator may notice even brief peaks – yet not be disturbed by meter flickering. Very short peaks, which may cause problems in *digital* transmission and storage systems, are underestimated in the original PPM due to the response time, so for peak measurements in the digital domain, the peak sample value is measured [7].

Although the PPM was not designed for – and not really suitable for – loudness measurement and alignment, some rules can be made to help an operator use the PPM for that purpose anyway [8], sect. 5.2. A major disadvantage of these rules is that they require knowledge of the actual type or genre of the source material. The standards describe different appropriate display scales, their contrast, brightness, colour etc. The human factor is taken into account in the specifications for decay time and peak-hold time. Meter scales for both mechanical and opto-electronic displays have been specified. Figure 2 shows one of the scales for the mechanical display; note the linear dB-scale.

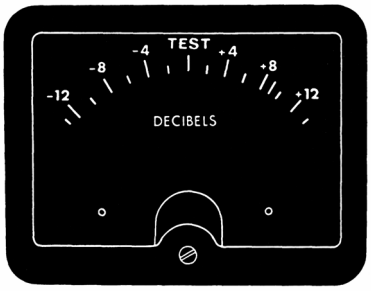


Figure 2: *PPM scale from [6].*

The opto-electronic display, as shown in Figure 3, features a non-linear correspondence between length of the bar and dB, but different from the VU meter with its linear voltage scale. Instead the bar-type of PPM takes advantage of the digital technology and adapts the scale graduation to the needs of the users, by providing a fine resolution at high levels *and* a large dynamic range. A minimum of 100 segments are specified for the bar-type instrument in order to give a smoothly changing length of the bar.

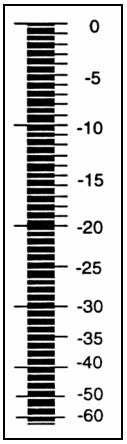


Figure 3: *Bar-type PPM scale from [7].*

The displays of traditional level meters, as described above, were determined by a mixture of technical and human factors, some of which have a scientific basis, whereas others are based on experience from their application domain. Although none of these meters are particularly suitable for measuring loudness, parts of their display properties have been applied to our presented loudness meter and associated display described in section 3.

2.2. Loudness meter standardisation within the ITU

Measurement algorithms and display requirements for loudness and true peak level meters have recently been described in the ITU recommendations [1] and [2]. Although the algorithms specified may not be the best ones available they have now been standardised so that new meters, providing a better estimate of the perceived loudness than a VU or PPM meter, can be made. In fact, the loudness measurement specified in [1] is not really measuring loudness, but rather an estimate of the *gain offset* required to match the loudness of one sound clip to that of a reference sound.

Due to the non-linear aspects of hearing, this gain offset and the corresponding change in loudness can differ. This issue is recognised in the recommendation. For operational purposes, however, the gain offset can be quite useful, as the operator has gain adjustment tools readily available.

The loudness measurement algorithm consists of a frequency-weighted RMS value, developed with measurement time intervals in the order of 10-30 seconds, i.e. an *Leq* (equivalent level). In the case of a multi-channel input, a single loudness value is computed based on a weighted energy-sum. A measurement period of 10s of seconds is long-term rather than short-term, and not directly suitable for a real time meter. But as no reference data for continuously varying loudness matching was available, the accuracy of a short-term measurement could not be tested.

Three different displays have been specified in [2], one of which is depicted in Figure 4. Compared to the VU and PPM meters notice the relatively few segments and the linear scale in LU (Loudness Units – equivalent to dB). Furthermore, the range of the scale is rather small – which is in accordance with the primary purpose of the meter as an aid in aligning loudness, and not a general-purpose level meter.

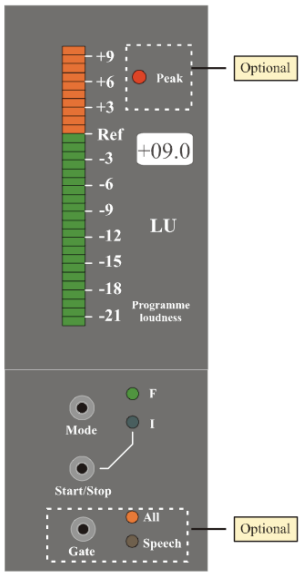


Figure 4: *Loudness meter display, according to [2].*

Having followed the ITU process of creating these two recommendations closely since 2003, we would like to make a few remarks: The contents of recommendations are not always that of the most solid science – but rather a combination of science, personal (or institutional/company) interests and political negotiations. On the positive side, [1] contains an opening for improved loudness measurement algorithms which are not as easily fooled as the simple frequency-weighted energy measure. Unfortunately, the statistics describing the results are insufficient to enable a scientifically valid comparison between the recommended method and alternative methods. Information on the statistical uncertainty, the variability of the listeners etc. are missing [9, 10]. Furthermore, the standard [1] calls for a short-term loudness measurement algorithm – which is certainly needed for real-time metering (and control).

3. A NOVEL LOUDNESS METER

In the loudness meter that we present here, we have employed successful visualisation principles of previous level meters, in combination with a new type of visualisation of long-term loudness history. The meter displays the short-term loudness, and the long-term loudness over a period of time, together with signal and overload indications. Our design of the meter's measurement algorithms and display has evolved as a mixture between science, intuition and empirical experience.

3.1. Measurement Algorithms

3.1.1. Loudness measurement

Loudness, as such, is a perceptual property of sound but can be modelled using different algorithms – and can thus be measured as an objective property of the sound.

Research into psychoacoustic models of loudness perception has been taking place for decades. Most prominently, Zwicker's loudness model has been standardised as ISO-532B [11]; however these models were developed for measuring loudness of sounds with stationary properties, such as noise and tones, and are thus unsuitable for meter applications [10]. More recently, Glasberg and Moore have presented research on modelling time-varying loudness of certain classes of signals [12, 13]. For loudness meter applications, simplified measurement algorithms have been developed, e.g. [14].

The present meter measures the loudness of the input signal by means of a simple model of loudness perception, but does not require any particular loudness model. For multi-channel input signals, a single loudness measurement is computed, combining the contribution from each channel.

The loudness meter uses the measurement unit of LU (Loudness Units). The LU is a measurement in dB, with 0 LU corresponding to a reference loudness level. The reference loudness level, and the acceptable range of fluctuation around it, might depend on the policy concerning the particular broadcast channel that the meter is monitoring.

In our meter prototype, the *TC LARM* algorithm [10] was employed, although other algorithms could alternatively be used. The accuracy of the TC LARM algorithm was evaluated in [10], against a set of subjective reference data, using a wide selection of speech and music material. Compared to the weighted Leq measurement algorithm in [1], the accuracy of TC LARM was found to be at least as good.

The short-term and long-term loudness measurements that the meter displays, use the same underlying measurement algorithm. However, the length of the analysis window and the visualisation of the measurements, differ for the short-term and long-term loudness. The temporal properties of the short-term loudness measurement were developed with several practical criteria in mind. For example: How much does the short-term loudness drop, in the 'silent' periods that are present in normal speech? How much does the short-term loudness rise, at the snare drum beat in pop/rock music? Similarly, the long-term measurement was developed to provide a constant measurement – within plus/minus a couple of LU – for material with an overall constant perceived loudness (see also section 3.2.4). In the current implementation of our loudness

meter prototype, we used the following lengths of the analysis windows: Short-term loudness: 0.5 s; Long-term loudness: 2.5 s.

3.1.2. Peak level

It is well established [15, 1] that the true peak value of a digital signal may be significantly above the magnitude of the actual samples. Especially signals that have been clipped or otherwise processed nonlinearly exhibit this property. When staying within the digital domain, and performing no subsequent processing, this will pose no problems (except from the distortion inherent in the non-linear processing). When changing domain or sample rate, however, the true – and possibly higher – peak value can appear in the new domain. That way, overload and additional audible distortion can result.

The peaks not directly represented by the samples can be estimated easily and accurately by using an interpolation technique, as used in oversampling and D-to-A conversion [1]. A short (FIR) lowpass filter near the Nyquist frequency and an interpolation factor of 4-8 will yield good estimates of the true peak value. Such a technique is employed in the presented meter.

In our loudness meter prototype, only the essential overload- and "signal present"-indications are displayed, for each input channel.

3.2. Visualisation of Measurements

3.2.1. Short-term loudness display

Figure 5 shows the display of the developed loudness meter, in a greyscale version (for better printing). In the loudness meter, a circular bar graph displays the short-term loudness of the input signal, as measured by the underlying loudness algorithm. This bar graph corresponds to the 'current' loudness that a listener would perceive. If the sound has a fairly constant loudness, the bar graph remains fairly constant (as opposed to traditional meters, that might change as a function of signal *amplitude* alone). The acceptable region of loudness is colour-coded: If the short-term loudness of a programme segment remains within the green region then the operator can easily determine that the material was neither too loud nor too soft.

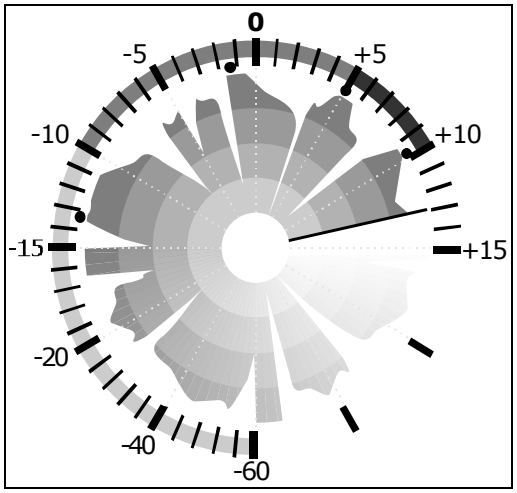


Figure 5: *A greyscale version of the developed loudness meter display.*

The range of the short-term loudness scale is larger than in the ITU meter, cf. Figure 4. The upper region from -20 to +15 LU use a linear scale, similar to the ITU recommendation. The scale tics in the upper region corresponds to the minutes on an analogue clock, and depending on the technology used, a display resolution of 0.1-0.5 LU is achievable. This resolution enables a smoothly changing display without perceivable flicker. The lower-level region of the display can be useful as a more detailed "signal present" indication, like in the PPM bar-type display (Figure 3).

Several properties of the short-term loudness display are in accord with knowledge about human visual perception, e.g., chapters 6 and 8 in [16]. One such property is *redundancy*, which increases the robustness of the readings. The short-term loudness is signalled not only by a position (of the end of the arc/curved bar) but also the size of the arc, i.e. the angle covered. Furthermore, the end of the arc changes angle according to the present loudness. As the eye is more sensitive to *angular movement* than to 'linear' movement this increases the readability of the display. Finally, *colours* are used to code relevant regions. Together, these properties also help ensuring that the short-term loudness is evident, even if a human operator was presented with several simultaneous displays, or if reading the meter from a distance.

3.2.2. Long-term loudness display

The long-term loudness is displayed in the centre of the loudness meter, by means of a *circular envelope graph*. The envelope graph is organized according to an absolute time-scale, similar to familiar analogue clocks. Thereby, the long-term loudness of the input signal during the past hour is displayed at all times. In Figure 5, the time is 9:13, hence the current long-term loudness is being displayed at the "13 minutes past the hour" position. The 'older' the long-term loudness entry is, the more it is faded into the background (white in Figure 5, black in Figure 6). Thus, the long-term loudness display appears like a "radar display" which is scanning clock-wise.

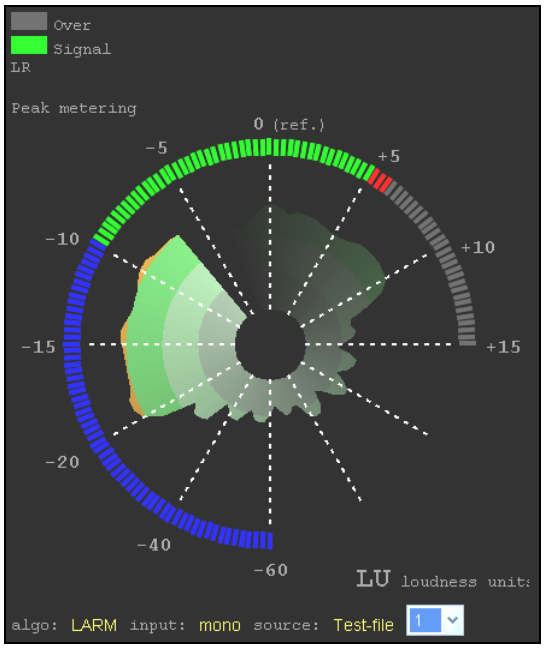


Figure 6: *A colour version of the developed loudness meter display, with a mono input signal.*

The further away from the centre the long-term loudness graph is, the louder the sound was at that time. The different colour regions correspond to "below reference loudness", …, "above reference loudness" (Figure 6).

As an extra feature, the displayed long-term loudness could cover any period of time, from the past minute (time is zoomed in) to the past 12 hours (time is zoomed all the way out). Furthermore, a single loudness meter could record or log the *loudness history* of several simultaneous sources, which the operator could then switch between while monitoring the sources.

We have implemented the loudness meter as a virtual prototype (software). Figure 6 shows the display of the prototype.

3.2.3. Signal level and Overload display

The loudness meter's signal level indicator consists of a column of LED-type components for each audio channel of the input source. The red indicator is turned on, for a short period, when signal overload is detected. Typically overload is indicated when the level is close to or above 0 dBFS, but lower levels could alternatively be used, depending on policy concerning the particular source. The green indicator is turned on when the signal level is above (say) -50 dBFS, to indicate that a signal is present on the corresponding channel – conversely to indicate a signal drop-out.

In addition, the signal overload indicator could display, using a coloured dot, any signal overloads (on any channel) that have occurred during the past hour (Figure 5). These dots are displayed along the circumference of the long-term loudness display, where their locations are used to indicate that certain events occurred at the corresponding time. Different colours could even be used to indicate (other) technical problems that happened in the past, such as signal drop out, or loss of clock sync for a digital input.

3.2.4. Example

Although a loudness meter is inherently an instrument to be used in real-time, for purposes of illustration the loudness measurement data can be extracted and plotted. The three graphs in Figure 7 show the short-term and long-term loudness measurements and the signal amplitude, as a function of time, using a test signal as input to the meter. This demonstrates how (much) the two loudness measures fluctuate for different types of audio material.

A test signal was made up of two audio segments, each of 15 seconds duration, representing characteristic signals: speech and pop music (Table 1). Whereas the speech signal is a fairly 'dry' recording, the pop track has undoubtedly been processed with dynamics compression and other mastering techniques. Each of the segments were 'level-normalised' individually, i.e. scaled to peak at 0 dBFS.

In test signal	CD	Track	Start time	End time
0-15 s	EBU SQAM: Sound Quality Assessment Material	#49: Speech - Female, English	0:00	0:15
15-30 s	Madonna: Confessions On A Dancefloor	#1: Hung Up	0:45	1:00

Table 1: *The contents of the test signal, displayed in* Figure 7.

The *short-term loudness* graph reveals that the speech segment consists of two spoken sentences, but also that the speaker achieves a fairly constant loudness, with variations within +/− 5 LU. Even though both segments were peak-normalised, the *long-term loudness* graph shows that the pop segment is consistently louder that the speech segment. In fact, the pop segment manages to stay 6-8 LU above the reference loudness level, with virtually no variation.

Imagine these two segments had been broadcast on the radio, with no further processing, while the loudness meter was 'listening' to the signal. The meter's long-term loudness history could then suggest that the pop segment should have been attenuated by say 5 dB, in order to spare the listeners for a noticeable increase in loudness. Lund describes the loudness meter's application in the context of broadcast for Digital TV and other media [17].

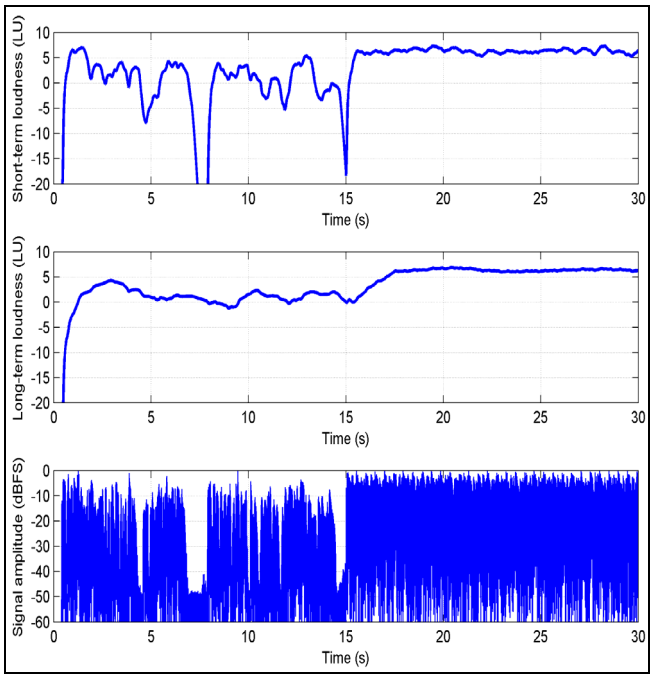


Figure 7: *The short-term loudness, long-term loudness and amplitude of a test signal consisting of 2 segments. First 15 sec: Female speech; Last 15 sec: Madonna - Hung Up.*

4. TOWARDS EVALUATION

The effectiveness of any loudness meter will depend on both the graphical appearance and dynamic behaviour of its display, as well as on its underlying measurement algorithms. All of these factors must be taken into account, when assessing the meter's overall quality and usability.

Formal evaluation of a visualisation system, such as the one described in this paper, is challenging: First of all, one or more metrics must be defined by which the display should be evaluated. The correspondence between the sound heard and the picture seen is one aspect to be evaluated. Another metric could characterise the speed of reading the meter reliably. A very high-contrast and flashing meter could cause eye fatigue (even though such a display might be immediately more readable). The usefulness of having several types of loudness measurements available at one glance may be hard to measure directly, but has to do with the compactness of the display, which again determines where the display can

fit in the application, and how many independent sources can be displayed on individual meters, in a given workspace.

So far our loudness meter has only been verified informally – we have received positive response from potential users in different application areas. No systematic experimental assessment of the proposed meter has been attempted yet.

4.1. An example of an evaluation method

A method for designing and evaluating a short-term loudness meter has been proposed in [18]. The purpose of that study was to address the need expressed by the ITU in [1]. At least two challenges are described in [18]: First, a way of creating a continuously varying measure of the perceived loudness must be found – i.e., a set of reference data for evaluating the measurement by the meter. Second, these time-varying reference data must be compared to the meter display. As a result of the evaluation, the technical parameters of the meter (or its measurement algorithm) can be set to appropriate values.

Rather than tracking the time-varying loudness itself, the task of the subjects in [18] was to continuously adjust a gain control to keep the loudness constant, that is, a continuously varying gain correction factor was registered. A couple of difficulties using this method were found: 1) When subjects adjusted the gain they tended to overshoot a bit. This must be taken into account when analysing the data. 2) The subjects tended to drift in their loudness reference. This means that their gain correction factor for *identical* sound segments changed over time.

To evaluate variations of a short-term loudness meter, the output of the ITU loudness measurement algorithm using different lengths of the analysis time window was plotted against the subjective gain adjustment data. The evaluation consisted of a visual inspection of these plots, and based on that a time window of 3 seconds was chosen as optimum for a "short-time loudness" measurement.

4.2. A proposed evaluation method

A loudness meter with its underlying measurement algorithms and display methods contains many parameters – more than could easily be adjusted in a traditional adjustment-type of experiment. Furthermore, the task of evaluating the complete meter in an experiment would require a considerable amount of time, as the inclusion of a signal history depends on listening to several sound segments within a single session.

One way to overcome the difficulties of performing a multi-parameter adjustment experiment would be to present a number of different *complete* meters, with pre-set variations of the underlying algorithms and their parameters, and maybe even display types. The task of the test subjects would in that case be to *rate* the different meters according to specific criteria (as the metrics mentioned above), as well as the subjective overall impression.

5. CONCLUSION

We have proposed some design criteria for visualising loudness features of an audio signal, measured along different time scales. We then presented a novel loudness meter, simultaneously showing three time-varying features of an audio signal: short-term loudness, long-term loudness history and a overload indicator. Our meter displays the short-term loudness using a circular bar graph. The long-term loudness is displayed by means of a circular envelope graph, organized according to an absolute time-scale – looking similar to a radar display. The presented real-time loudness meter thus provides one complete solution to the requirements for an effective loudness meter. The algorithms underlying the meter prototype's loudness and peak-level measurement take into account recent ITU-R recommendations and research into loudness modelling. Finally, different aspects of evaluating a loudness meter were discussed.

6. REFERENCES

[1] ITU-R (2006) "Rec. ITU-R BS.1770, Algorithms to measure audio programme loudness and true-peak audio level.", International Telecommunications Union.

[2] ITU-R (2006) "Rec. ITU-R BS.1771, Requirements for loudness and true-peak indicating meters", International Telecommunications Union.

[3] Emmett, J. (2003) "Audio levels - in the new world of digital systems", EBU Technical Review, vol.2003:January.

[4] Moerman, J.P. (2004) "Loudness in TV Sound", in Proc. of the AES 116th Conv.

[5] IEC (1990) "IEC 268-17. Sound system equipment - Part 17: Standard volume indicators", International Electrotechnical Commission.

[6] IEC (1991) "IEC 268-10. Sound system equipment - Part10: Peak programme level meters", International Electrotechnical Commission.

[7] IEC (1995) "IEC 268-18. Sound system equipment - Part 18: Peak programme level meters - Digital audio peak level meter", International Electrotechnical Commission.

[8] Dickreiter, M. (1987) "Handbuch der Tonstudiotechnik, Band 1" (5. Auflage. ed.), München: K. G. Saur.

[9] Skovenborg, E., Quesnel, R. & Nielsen, S.H. (2004) "Loudness Assessment of Music and Speech", in Proc. of the AES 116th Convention, Berlin.

[10] Skovenborg, E. & Nielsen, S.H. (2004) "Evaluation of Different Loudness Models with Music and Speech Material", in Proc. of the AES 117th Convention, San Francisco.

[11] ISO (1975) "Acoustics. Method for calculating loudness level. International Standard ISO 532 (1.ed.)", International Organisation for Standardisation.

[12] Glasberg, B.R. & Moore, B.C.J. (2002) "A Model of Loudness Applicable to Time-Varying Sounds", Journal of the Audio Engineering Society, vol.50:5, pp.331-342.

[13] Moore, B.C.J., Glasberg, B.R. & Stone, M.A. (2003) "Why Are Commercials so Loud? -- Perception and Modeling of the Loudness of Amplitude-Compressed Speech", Journal of the Audio Engineering Society, vol.51:12, pp.1123-1132.

[14] Jones, B.L. & Torick, E.L. (1982) "A New Loudness Indicator for Use in Broadcasting", in Proc. of the 71st AES Convention, Montreux.

[15] Nielsen, S.H. & Lund, T. (2003) "Overload in Signal Conversion", in Proc. of the 23rd AES Intl. Conf.

[16] Goldstein, E.B. (1989) "Sensation and Perception" (3rd. ed.), Belmont: Wadsworth Publishing Company.

[17] Lund, T. (2006) "Control of Loudness in Digital TV", in Proc. of the NAB-2006 Convention.

[18] Soulodre, G. & Lavoie, M.C. (2006) "Development and Evaluation of Short-Term Loudness Meters", in Proc of the 121st AES Conv.

THE ORIGINS OF DAFX AND ITS FUTURE WITHIN THE SOUND AND MUSIC COMPUTING FIELD

Xavier Serra

Music Technology Group
Universitat Pompeu Fabra
Barcelona, Spain
xserra@iua.upf.edu

ABSTRACT

DAFX is an established conference that has become a reference gathering for the researchers working on audio signal processing. In this presentation I will go back ten years to the beginning of this conference and to the ideas that promoted it. Then I will jump to the present, to the current context of our research field, different from the one ten years ago, and I will make some personal reflections on the current situation and the challenges that we are encountering.

1. ORIGINS OF DAFX

The International Conference on Digital Audio Effects is celebrating its 10th anniversary. None of the researchers that more than ten years ago were involved in the creation of the conference would have ever expected to be celebrating this anniversary. In the name of the promoters of DAFX I would like to thank all the participants to these conferences for supporting it and especially I want to congratulate the organizers of these ten meetings for making it happen.

The DAFX was started as part of a European project for cooperation and scientific transfer, named Digital Audio Effects, which lasted from 1997 to 2001. The project was coordinated by the French researcher Daniel Arfib and its main objective was to provide a synthesis of what can be done in the digital processing of sounds, and its application to music. The project resulted in two successful initiatives, the DAFX book [1] and the DAFX Conferences (more information can be found in http://www.dafx.de/).

The DAFX book, edited by Udo Zölzer, came out in 2002 and in a short time it became a major reference. It covers the main topics of digital audio effects, such as the basics for digital filters, modulations, non-linear processing, spatial effects, the more advanced topics in audio processing based on time-segment, time-frequency, source-filter, spectral analysis, time-frequency warping and also a topic on control issues and another on the new techniques of bitstream processing. One of the main reasons for its success lies in its practical approach and in all the MATLAB code, which makes it easy for someone getting into the filed to try out the algorithms while understanding the theory behind them. Udo Zölzer did a great job in promoting a unified style from all the contributors and also in carrying out a careful overall editing.

The DAFX conference was started as an international meeting of researchers interested in the theory and practice of digital sound processing and its applications. The goal of the conference was to offer both an overview of the field and an in-depth discussion of current research and future directions. The first scientific committee of the conference included the partners of the EU project. I became the chairman of the DAFX-98 in Barcelona, and since then the chairmen have been: Jan Tro (DAFX-99, Trondheim), Davide Rocchesso (DAFX-00, Verona), Mikael Fernström (DAFX-01, Limerick), Udo Zölzer (DAFX-02, Hamburg), Mark Sandler (DAFX-03, London), Gianpaolo Evangelista (DAFX-04, Naples), F. Javier Casajús (DAFX-05, Madrid), Philippe Depalle (DAFX-06, Montreal) and Sylvain Marchand (DAFX-07, Bordeaux). Each organizer has taken the previous conference a step further and thus this year conference is quite different from the first one.

Going back to the beginning, the first major discussion within the scientific committee was in defining the scope of the term Digital Audio Effects. With a narrow perspective it would only include what is commercially known as "Digital Audio Effects Processors" and the technologies behind them. But we found it more appropriate to widen the perspective, including all the digital processes that have a sound as input and their output is a signal useful for audio and music applications. We decided that topics that could be covered in the conference included: Filtering, Modulation, Delay, Non linear processing, Time/Frequency scaling, Spatialisation, Sound analysis, Spectral processing, Audio coding, Hardware, and Software implementations. We were particularly interested in the new research development, that are extending the traditional low level sound processing found in most commercial products towards higher level processing techniques, techniques that could be described by the term "Content Based Processing".

Looking at the call for papers and the final proceedings of all the ten conferences we can observe that it has evolved. The scope of the conference has expanded, new topics have been incorporated and new challenges are being tackled. It is important that DAFX reflects the state of the art of the field and it continues to evolve by being sensitive of its context.

2. SOUND AND MUSIC COMPUTING CONTEXT

The topics being presented and discussed at DAFX can be considered part of what is now called Sound and Music Computing (SMC). A good overview of this research field is the Roadmap funded by the EU and elaborated by the S2S2 consortium [2]. The Roadmap document covers quite a number of issues but one of the main contributions is the definition of the actual field of research. In the Roadmap it is stated that the SMC research approaches the whole sound and music communication chain from a multidisciplinary point of view, and that by combining scientific,

technological and artistic methodologies it aims at understanding, modeling and generating sound and music through computational approaches.

The sound and music communication chain covers all aspects of the relationship between sonic energy and meaningful information, both from sound to sense (as in musical content extraction or perception), and from sense to sound (as in music composition or sound synthesis). The disciplines involved in SMC cover both human and natural sciences. Its core academic subjects relate to musicology, physics (acoustics), engineering (including computer science, signal processing and electronics), psychology (including psychoacoustics, experimental psychology and neurosciences) and music composition. Most SMC research is quite applied and current areas of application include digital music instruments, music production, music information retrieval, digital music libraries, interactive multimedia systems, auditory interfaces and augmented action and perception (e.g. bionic ears, digital prosthesis and multimodal extensions of the human body).

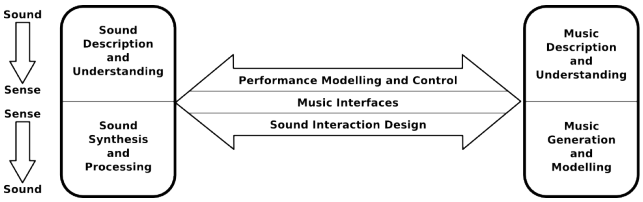


Figure 1: *SMC research areas (from [3]).*

Figure 1 depicts the relationships between the different SMC research areas. It makes a basic distinction between research that focuses on sound (left hand side), research that focuses on music (right hand side) and the research fields that address the interaction between the two. For each research field, there is an analytic and a synthetic approach. The analytic approach goes from encoded physical (sound) energy to meaning (sense), whereas the synthetic approach goes in the opposite direction, from meaning (sense) to encoded physical (sound) energy. Accordingly, analytic approaches to sound and music pertain to analysis and understanding, whereas synthetic approaches pertain to generation and processing. In between sound and music, there are multi faceted research topics that focus on interactional aspects. These are performance modeling and control, music interfaces, and sound interaction design.

DAFX covers quite a big part of the SMC field, and together with the International Computer Music Conference (ICMC), the International Conference in Music Information Retrieval (ISMIR) and the International Conference on New Interfaces for Musical Expression (NIME), DAFX represents quite well the core of the SMC research community.

The development of audio systems cannot be approached just from a signal processing perspective and the needed inderdisciplinarity approach to solve the problems discussed at DAFX is quite clear. The disciplines involved in SMC range from the natural sciences like physics and acoustics through mathematics, statistics and computing, all the way to physiology, psychology and sociology. The increased recognition of SMC and of the multidisciplinarity fields in general should help DAFX to position itself as a research conference with a specific personality.

3. SOME CHALLENGES

The SMC Roadmap identifies two broad research challenges: (1) To design better sound objects and environments and (2) To understand, model, and improve human interaction with sound and music.

The first challenge relates to the fact that many current electronic devices, not just the audio ones, incorporate sound systems in them and thus the improvements in the sounds produced by all these devices present in our environment will enhance our quality of life. Thus our society will benefit from the development of new musical instruments, new technologies for delivering sounds, new sound modeling strategies, new sonic spaces, an also from a better control of the environmental sound and its pollution consequences. The DAFX community should be proactive in developing the core technologies to face this challenge.

The second challenge is concerned with the issue that truly useful and rewarding machine-mediated sonic environments and services will require a better understanding of human interaction with sound and music in all its breadth, including perceptual, cognitive, emotional, bodily and social aspects. We need to develop computational models of auditory perception and cognition, new perception paradigms and technologies for bridging the semantic gap in music. We also need to better understand the expressive issues of sound communication and the relation between perception and action.

The SMC Roadmap also identifies more contextual challenges of relevance here. Beyond the research issues, the DAFX community needs to worry about education, social aspects and also about technology transfer. The increasing need for specialists in our field requires a decisive growth in the size and quality of existing educational programs and the creation of appropriate new ones. Thus we need to appropriately educate our future researchers. Also social concerns have to play an important role in our research decisions. For example we must be able to empower users, putting the relevant choices and decisions into the hands of the individual. Finally the improvement of the technology transfer requires especial efforts. A large part of our research is devoted to applications that can be directly exploited in the arts, in industry and in society at large. Proper knowledge transfer can lead to successes whose size and impact are bound to be very large.

4. CONCLUSIONS

DAFX and its community have played an important role in the development of the Sound and Music Computing field and it definitely has the potential for continuing to do so. This 10th anniversary is a good moment to reflect on what we are, on our role within the larger research community and on the challenges that we have in front of us. I have tried to give my personal view on this and I just hope to have triggered some discussion within our research community.

5. REFERENCES

[1] Zölzer, Udo. Ed. *DAFX - Digital Audio Effects*. John Wiley & Sons, 2002. ISBN: 0-471-49078-4

[2] X. Serra; M. Leman, and G. Widmer, Eds., *A Roadmap for Sound and Music Computing*. 2007 ed. The S2S Consortium. http://www.soundandmusiccomputing.org/roadmap/.

MODAL PARAMETER TRACKING FOR SHAPE-CHANGING GEOMETRIC OBJECTS

Cynthia Bruyns Maxwell
University of California at Berkeley
Computer Science Department, CNMAT
Berkeley, California USA
cbruyns@cs.berkeley.edu

David Bindel
New York University
Courant Institute of Mathematical Sciences
New York, New York USA
dbindel@cims.nyu.edu

ABSTRACT

For interactive sound synthesis, we would like to change the shape of a finite element model of an instrument and rapidly hear how the sound changes. Using modal synthesis methods, we would need to compute a new modal decomposition with each change in the geometry, making the analysis too slow for interactive use. However, by using modes computed for one geometry to estimate the frequencies for nearby geometries, we can hear much more quickly how changing the instrument shape changes the sound. In this paper, we describe how to estimate resonant frequencies of an instrument by combining information about the modes of two similar instruments. We also describe the balance between computational speed and accuracy of the computed resonances.

1. INTRODUCTION

With fast computers and modern techniques, we can synthesize realistic instrumental sounds in real time. The goal of the work we describe here is to *design* realistic instruments in real time. That is, we would like to know precisely how changing the shape of an instrument, or the materials that make up the instrument, will change the instrument's sound.

In modal synthesis, the motion of an instrument is expressed as a combination of modes, each of which oscillates independently of the others. To use modal synthesis, though, we must first compute a partial eigenvalue decomposition of the system matrices. This eigenvalue problem is relatively expensive, but we only need to compute the decomposition once for a given instrument. However, the eigenvalues and eigenvectors depend strongly on the instrument's shape. Therefore, to design new instrument shapes with standard modal analysis, we would need to recompute the modes for each new design – a prohibitively expensive step for an interactive tool. Our goal in this paper is to show how to quickly estimate the modes of a new instruments from the modes of one or more similar instruments.

This paper describes one method that can be used to predict how the eigenvalues and eigenvectors will move when the geometry changes. The method exploits properties of parameter-dependent linear systems by tracking an invariant subspace as modifications are made. Using this method, one avoids recomputing modes while still providing an accurate representation of the timbre of an object. The results show very high accuracy for moderate changes. Moreover, our algorithm runs in a modest linear time for standard finite element discretizations.

1.1. Mathematical preliminaries

For a system of *n* degrees-of-freedom (DOFs), the governing equations of motion are a set of *n* coupled ordinary differential equations of second order. The solution of these equations becomes complicated when the size of the system is large or when the forcing functions of the system are non-periodic. In such cases, it is convenient to express the deformation of the object as linear combinations of normal modes of the system. Such a transformation uncouples the equations of motion into a set of *n* uncoupled differential equations. In this form it is trivial to solve for the vibration of an object under various loading conditions.

Even when the motion of the object is large, or other nonlinear behaviors occur that violate the assumptions of modal superposition, the techniques presented in this paper can be used to build the basis that captures object motion [1], [2]. As such, this paper provides a general technique for approximating modal parameters as objects undergo shape change. It is used to aid in modal decomposition and is applied before excitation and modal superposition are performed.

The canonical system of equations resulting from discretization using the finite element method is as follows:

$$M\ddot{u} + C\dot{u} + Ku = f(t) \quad (1)$$

where M is the matrix representing the distribution of mass in the system, C is a measure of damping and K is the stiffness matrix. This equation expresses the balance of forces generated by the acceleration, velocity and displacement of the object. In this form the equations are coupled and thus the solution involves manipulation of these large system matrices.

Alternatively, modal analysis seeks to decouple this system into single degree-of-freedom (DOF) oscillators. Without damping, the procedure for uncoupling these equations is straight forward using the general eigenvalue decomposition $Kx = \lambda Mx$. However, with damping, decoupling these equations requires some assumptions to be made [3]. Normally proportional damping is assumed such that:

$$C = \alpha_1 M + \alpha_2 K \quad (2)$$

Substituting this expression back into Equation 1, we have:

$$M(\ddot{u} + \alpha_1 \dot{u}) + K(\alpha_2 \dot{u} + u) = f(t) \quad (3)$$

This is the general form of the system before eigendecomposition.

1.2. Model reduction

The eigenvalue problem that we want to solve then is:

$$Ax = \lambda Bx \quad (4)$$

where A and B are the positive semi-definite symmetric stiffness and mass matrices respectively (i.e. K and M), and x is the vector of nodal displacements of the mode with natural frequency $\lambda = \omega^2$. One means of formulating approximate equations for freely vibrating discrete systems is via the Rayleigh's quotient:

$$\lambda_R = \frac{\hat{x}^T A \hat{x}}{\hat{x}^T B \hat{x}} \tag{5}$$

where $\hat{x}$ is an approximation to x [4]. The relative accuracy of methods based upon this formulation results from the fact that eigenvalues λ are stationary with respect to perturbations in the elements of A, B, and the eigenvectors x. Thus, if a transformation for the n physical node displacements, $\hat{x}$, into fewer $(m < n)$ generalized coordinates is available, say

$$\underset{n \times 1}{\hat{x}} = \underset{n \times m}{V} \underset{m \times 1}{y} \tag{6}$$

then the corresponding Rayleigh quotient becomes

$$\lambda_R = \frac{y^T V^T A V y}{y^T V^T B V y}. \tag{7}$$

Making λ_R stationary to arbitrary variations in the m elements of y yields the reduced eigenproblem

$$V^T A V y = \lambda_R V^T B V y. \tag{8}$$

We can view this reduction as imposing $n - m$ constraints on the original system thus giving the following result using the Cauchy Interlace Theorem

$$\lambda^{(i)} \leq \lambda_R^{(i)} \leq \lambda^{(j+n-m)} \;\; j \leq m. \tag{9}$$

Thus all the λ_R are contained between $\lambda^{(i)}$ and $\lambda^{(n)}$ and the approximations become exact for $m = n$.

The essence of the reduction scheme lies in the definition of the transformation matrix V. Some researchers have used matrices comprised from vectors that span a Krylov subspace [5], [4]; we choose to use a matrix that is made from exact modal vectors [6].

2. METHODS

For most systems, only the first few natural frequencies and associated natural modes greatly influence the dynamic response, and the contribution of higher natural frequencies and the corresponding mode shapes is negligible. If only the fundamental natural frequency of the system is required, the Rayleigh method can be used. However, if a *small number* of lowest natural frequencies of the system is required, the Rayleigh-Ritz method can be used.

The Rayleigh-Ritz method then, can be considered an extension of the Rayleigh method [7]. In the Rayleigh-Ritz method, the shape of deformation of the continuous system, $v(x)$ is approximated using a trial family of admissible functions that satisfy some geometric boundary condition of the problem:

$$v(x) = \sum_{i=1}^{n} c_i \phi_i(x) \tag{10}$$

where c_i are unknown constant coefficients and ϕ_i are the known trial family of admissible functions. The functions can be a set of assumed mode shapes, polynomials, or eigenfunctions.

The accuracy of the method depends on the value of n and the choice of trial functions $\phi_i(x)$ used in the approximation. By using a larger n, the approximation can be made more accurate, and by using trial functions which are close to the true eigenfunctions, the approximation can be improved.

2.1. Approximations from a subspace

Let s denote a geometric parameter. For a given finite element model, we have a generalized eigenvalue problem:

$$(K(s) - \lambda(s) * M(s))u(s) = 0, \tag{11}$$

where $K(s)$ is the stiffness matrix of the system and $M(s)$ is the mass matrix at the given state of the geometry, and $\lambda(s)$ and $u(s)$ are an eigenvalue and its corresponding eigenvector for the system.

If $w(s)$ is accurate to $O(h)$ as an estimate for $u(s)$, then

$$\mu(s) = (w(s)^* K(s) w(s))/(w(s)^* M(s) w(s)) \tag{12}$$

is accurate to $O(h^2)$ as an estimate for $\lambda(s)$. This is the accuracy boost we want to utilize.

Suppose that we have computed eigenpairs $(\lambda(s_0), u(s_0))$ and $(\lambda(s_1), u(s_1))$, and now want to compute the pair $(\lambda(s_2), u(s_2))$. Then we can use the initial approximation $\mu(s)$ drawn from a Rayleigh-Ritz approximation on the pencil:

$$(U^* K(s_2) U, U^* M(s_2) U) \tag{13}$$

where $U = [u(s_0), u(s_1)]$ (or if several of the lowest eigenvalues are desired then simply replace $u(s_0)$ with $u1(s_0), u2(s_0)$, ... and $u(s_1)$ with $u1(s_1), u2(s_1)$, etc.).

If the step size is $O(h)$, then the error in approximating $u_i(s_2)$ by extrapolating through $u_i(s_0)$ and $u_i(s_1)$ should be $O(h^2)$ – the approximation is good through the linear term – and the eigenvalue approximation should be $O(h^4)$. More generally, if you use invariant subspaces computed at k points, you should get $O(h^k)$ accuracy in the eigenvector, and a corresponding $O(h^{2k})$ accuracy in the computed eigenvalue.

Therefore, by building a basis from n eigenvectors sampled at k locations in parameter space, we can predict the same n eigenvectors and the corresponding eigenvalues at nearby points. In essence, by looking at a couple of steps, we can capture the behavior of the eigenvectors rotating as the geometry changes and by solving a smaller eigenproblem, we can reduce the time to compute the original system in order to determine a subset of eigenvalues and eigenvectors.

2.2. Expected behavior

Instead of solving the entire eigenvalue problem, we will be making approximations to the solutions by projecting onto the subspaces formed by analyzing nearby shapes. We can measure how well our method approximates the true eigenvectors of the system by measuring the angle between the actual and approximated eigenvectors [8], [6], [9]. One can approximate the angle by:

$$\| \sin \psi \| \leq \frac{\|r(y)\|}{gap(\theta)} \tag{14}$$

where $\theta = y^* A y$ is the Rayleigh quotient given by projection of the matrix A onto the vector y. The residual $r(y) = Ay - y\theta$ measures how well the vector y approximates an eigenvector. Also

the $gap(\theta) = min|\lambda_i[A] - \theta|$ over all $\lambda_i \neq \alpha_i$ measures how well-separated a given eigenvalue is. This result combines several important facts. Firstly, it says that the larger the gap between an eigenvalue and the neighboring eigenvalues in the same spectrum, the better the approximation one can make to its eigenvector. The next fact is more straightforward; it states that the better the vector y acts like a solution to the eigenproblem $Ax - \lambda x = 0$, the better it approximates an eigenvector. From these results we know that systems with repeated or tightly clustered eigenvalues will be a problem, and we will give an example on the effect of approximation techniques on these systems in Section 3.2. However, for general parameter dependent matrices, we can see that approximation using projections onto a subspace show promise.

We will try the Rayleigh-Ritz technique on several examples to see how well we can approximate the spectrum after modifications to the geometry.

3. RESULTS

We tested the usefulness of this approach on a variety of experiments. The geometries tested do not represent full instruments per-se, but instead are arbitrary shapes that can be formed using the parametric method described in [10]. We use these shapes for examination of the method.

For each geometry, we used a linear shell finite element formulation as described in [11]. Each element consists of four nodes each with six degrees-of-freedom. We use shells to make the axisymmetric example easier to visualize. However this method can be used with any geometric discretization that can be defined parametrically, including solid models.

3.1. Separated spectrum

First, we examined a shell whose curvature is defined by four control points as shown in Figure 1. The large dots indicate the points modified directly and the smaller dots represent the points which are interpolated using cubic B-spline interpolation to define the curve. By changing the location of these control points, we change

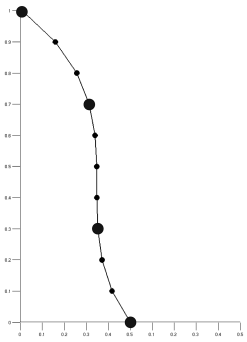


Figure 1: Parametric curve.

the geometry parametrically.

The control points define a curve which is then revolved by a small amount around the z-axis to form the geometry shown in Figure 2. We use this geometry to highlight one aspect of the method that occurs when the curve is revolved all the way around the z-axis.

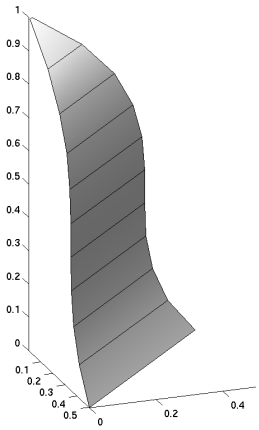


Figure 2: Simple shell model.

We examined changing a 1m tall by 0.5m radius shell's outermost radius by 10cm, 1cm, and 1mm and examined the error in eigendecomposition. We considered using only one sample point in parameter space, s, and examined the accuracy in prediction of the Ritz values. The results show prediction errors of 1% for the smallest step size and 100% for the largest step size. As expected,

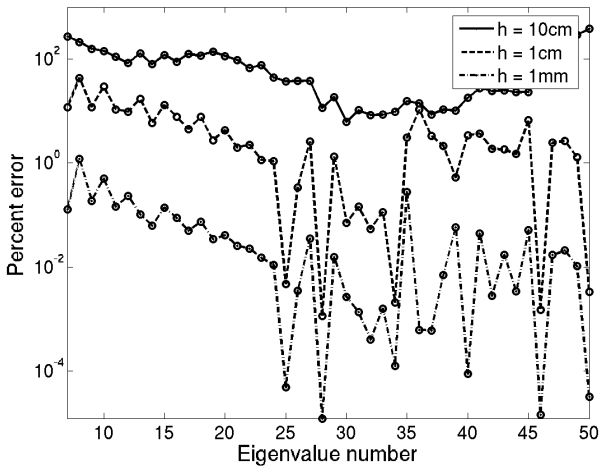


Figure 3: Error for different size h. One sample point.

smaller changes in geometry allowed for better prediction of the new eigensolution. In each of the plots, we consider the first n non-zero eigenvalues.

Next, we examined using two sample points. Figure 4 shows the results for the different step sizes. As expected, using more points in parameter space increased the accuracy of the predictions. In fact, the accuracy of the two-subspace version is almost

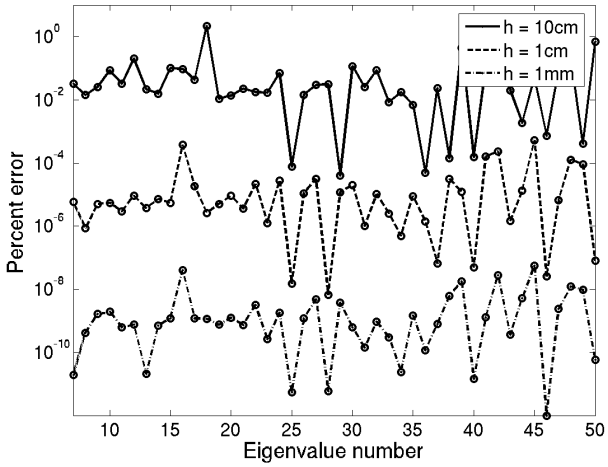


Figure 4: Error for different size h. Two sample points.

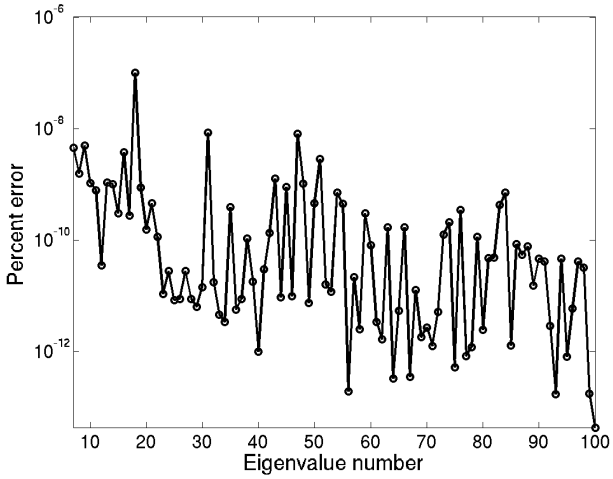


Figure 6: Error for $h = 10cm$. Larger subspaces.

twice as many digits as the one-subspace version which agrees with the theoretical bounds in Section 2.2.

Figure 5 shows that using two subspaces versus one also gives much faster convergence. Notice how the two point version has a steeper slope than the one point version, following the expected $O(h^{2k})$ convergence, (where k is the number of points).

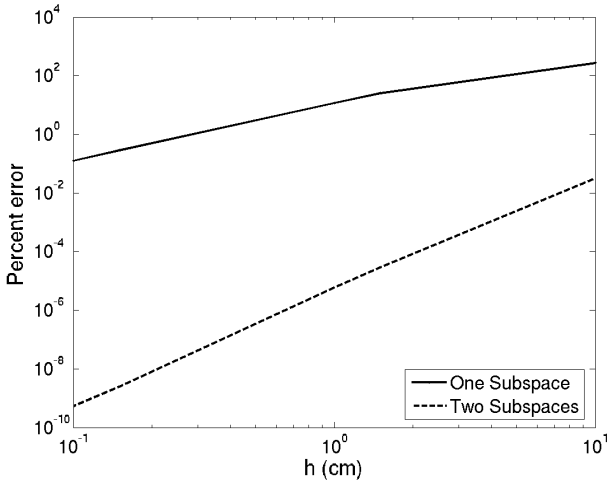


Figure 5: Error for different size h. Two sample points.

We also investigated using a larger subspace. So instead of using the first 50 eigenvectors, we used the first 100. Figure 6 shows how using more eigenvectors from each of the two subspaces improves the estimate of the eigenvalues.

We also confirmed that the error is proportional to the size of the object, i.e. making a 10cm change in a 1000cm object should produce smaller errors than the same absolute change to a 10cm object. By examining Figure 7, we can see when that the error is proportional to the size of the object, as expected.

3.2. Repeated eigenvalues and other difficulties

As we would expect for a model with a high degree of rotational symmetry, our test problem exhibits many repeated eigenvalues. For an eigenvalue with multiplicity m, we cannot uniquely identify m mode shapes. Even for nearly-symmetric objects, the mode shapes associated with a cluster of eigenvalues can vary wildly under small perturbations. Only the m-dimensional invariant subspace spanned by all the shapes for the eigenvalue cluster is uniquely defined.

The sensitivity of the mode vectors does not, on its own, imply that our method will behave poorly in the presence of repeated eigenvalues. If every vector in the invariant subspace for a cluster of m eigenvalues can be approximated well by some vector in the projection basis U, then we expect Rayleigh-Ritz approximation with U to produce a cluster of m eigenvalues near the original eigenvalues, and a corresponding subspace which is a good approximation to the true invariant subspace. However, the single-vector Lanczos iteration we use to find mode shapes sometimes fails to find a complete basis for each invariant subspace. When this occurs, we can overlook some of the eigenvalues and eigenvectors that we would like to capture in our projection space. When this occurs, the missing mode shapes represent a significant failure in our method.

For example, when analyzing the 1m high by 1m radius axisymmetric geometry shown in Figure 8 we found very large errors. Figure 9 shows the degeneracy that arises using a very axisymmetric geometric formulation.

We were able to resolve the some of the eigenvalues more accurately with a larger subspace, but not the eigenvalues at the beginning of the spectrum (see Figure 10).

The large errors at the beginning of the spectrum are not expected on their own, however when examining the eigenvalues we can see that the large errors can be attributed to the rapid change in eigenvectors for even a small perturbation. Figure 11 shows how the principle angles between the subspaces at two iterations can be very large for even small modifications.

In fact we found that the higher number of repeated eigen-

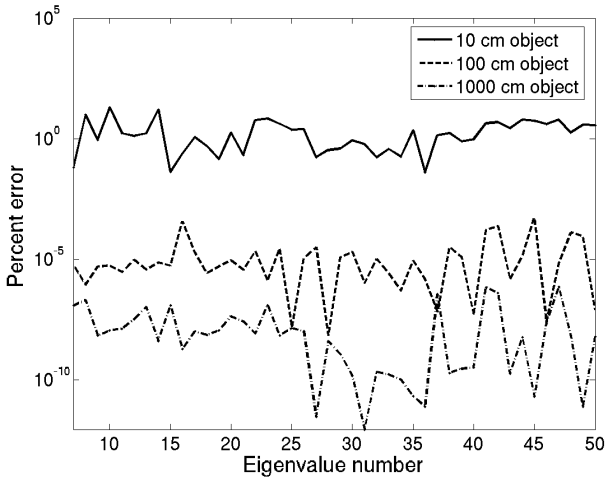


Figure 7: Error for different sized objects.

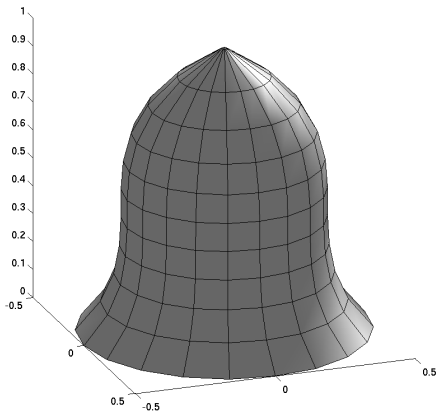


Figure 8: Axisymmetric shell model.

values, the worse the overall approximations. Figure 12 shows a geometry similar to Figure 8 but with 4 radial slices, and Figure 13 shows the error in eigenvalue approximations using this geometry. Figure 14 shows a geometry with 8 radial slices, and Figure 15 shows the error in eigenvalue approximations on this model. Figure 16 shows a geometry with 16 radial slices, and Figure 17 shows the resulting error in eigenvalue approximations. These results leads us to believe that more radial slices creates more blocks of symmetry in the system matrix, thus making it harder it is to approximate the motion of the eigenvectors.

We know two methods to address the difficulty of completely resolving the the invariant subspaces corresponding to repeated eigenvalues. The first method is to use a block version of the standard Lanczos algorithm [6]. Unlike the ordinary Lanczos iteration we use, block Lanczos iteration can find an invariant subspace for an eigenvalue in a single step, provided the block size is the same or greater than the multiplicity of the eigenvalue. The second method is the radial decomposition technique described in [10], which uses analytical knowledge of the symmetry group leading to the multiple eigenvalue in the first place.

Another source of problems is when a structure is much stiffer in some directions than in others. For example, our shell structure is much less resistant to out-of-plane bending than to in-plane compression. A vector that represents pure bending motion for one geometry may represent a mixture of bending and in-plane compression in a nearby geometry, so that a Rayleigh-Ritz approximation based on that vector will overestimate the frequency at the new geometry.

3.3. Performance

The speedup gained by using this method over traditional reanalysis is the difference between modest linear and super-linear computing time once the initial k samples have been computed. Figure 18 shows the speedup using this method over using reanalysis for increasing resolution of the object shown in Figure 2.

4. CONCLUSIONS

The aim of this investigation is to determine if our tracking method method can be used to predict the changes in the frequency spectrum of an object as parametric changes are made. The results of these experiments show that for moderate changes, it is possible to avoid recomputing the eigendecompositions in order to resolve the resonant frequencies of interest.

By exploiting the properties of the system matrices, we have a bound on the errors produced using different step sizes. For an interactive design tool, this would mean that the software could alert the user when errors above a given threshold have been made and signal the need for a full reanalysis.

For systems with many repeated eigenvalues, such as axisymmetric systems, it is more beneficial to use analysis techniques that will handle the multiple eigenvalue problem.

This investigation demonstrates that for interactive design applications, it is beneficial to track the spectrum for moderate changes in geometry to avoid computing a partial eigendecomposition. By using this method, we can maintain a moderate linear time algorithm with increasing system size.

5. ACKNOWLEDGEMENTS

Thanks to Carlo Séquin, Antoine Chaigne and Julius O. Smith III for their comments on this work.

6. REFERENCES

[1] K.A. Kline, "Dynamic analysis using a reduced basis of exact modes and Ritz vectors," *AIAA Journal*, vol. 24, pp. 2022–2029, 1986.

[2] M. Lou and G. Chen, "Modal perturbation and its applications in structural systems," *Journal of Engineering Mechanics*, vol. 129, no. 8, pp. 935–943, 2003.

[3] G. B. Warburton and S. R. Soni, "Errors in response calculations for non-classically damped structures," *Earthquake*

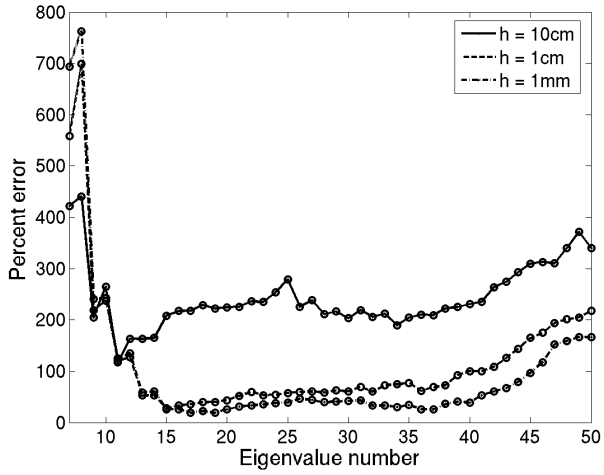


Figure 9: Error for different size h. Axisymmetric geometry.

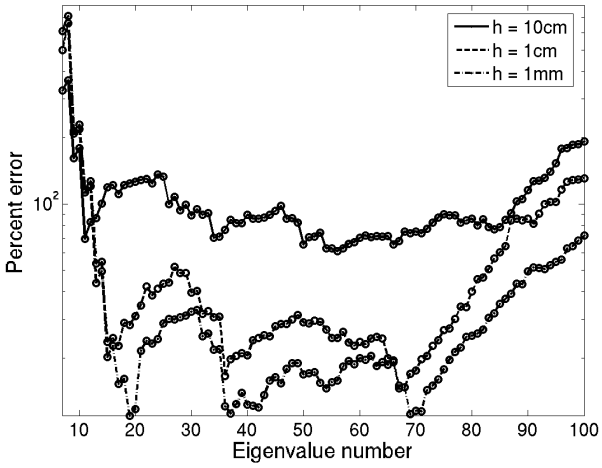


Figure 10: Error for different size h. Axisymmetric geometry. Larger subspace.

Engineering & Structural Dynamics, vol. 5, no. 4, pp. 365 – 376, 1977.

[4] I.U. Ojalvo and M. Newman, "Vibration modes of large structures by an automatic matrix-reduction method," *AIAA Journal*, vol. 8, no. 7, pp. 1234–1239, 1970.

[5] R.K. Kapania and C. Byun, "Reduction methods based on eigenvectors and Ritz vectors for nonlinear transient analysis," *Computational Mechanics*, vol. 11, pp. 65–82, 1993.

[6] B. Parlett, *The Symmetric Eigenvalue Problem (Classics in Applied Mathematics)*, SIAM, Philadelphia, 1998.

[7] S. S. Rao, *Vibration of Continuous Systems*, chapter 17, pp. 661–670, Wiley, 2007.

[8] C. Davis and W.M. Kahan, "The rotation of eigenvectors by perturbation III," *SIAM Journal of Numerical Analysis*, vol. 7, no. 1, pp. 1–46, 1970.

[9] M.E. Argentati, *Principal angles between subspaces as related to Rayleigh quotient and Rayleigh-Ritz inequalities with applications to eigenvalue accuracy and an eigenvalue solver*, Ph.D. thesis, University of Colorado at Denver, 2003.

[10] C. Bruyns and D. Bindel, "Shape-changing symmetric objects for sound synthesis," in *Proceedings AES San Francisco*, 2006.

[11] O. C. Zienkiewicz and R. L. Taylor, *Finite Element Method: Volume 2, Solid Mechanics*, Butterworth-Heinemann, Burlington, MA, 5th edition, 2000.

[12] J. W. Demmel, *Applied Numerical Linear Algebra*, SIAM, Philadelphia, 1997.

[13] R. L. Fox and M.P. Kapoor, "Rate of change of eigenvalues and eigenvectors," *AIAA Journal*, vol. 6, no. 12, pp. 2426–2429, 1968.

[14] T. Kato, *Perturbation Theory for Linear Operators*, Springer-Verlag, corrected printing of the second edition edition, 1995.

[15] Y.M. Ram, J.J. Blech, and S.G. Braun, "Eigenproblem error bounds with application to the symmetric dynamic system modification," *SIAM Journal on Matrix Analysis and Applications*, vol. 11, no. 4, pp. 553–564, 1990.

[16] Y.M. Ram and S.G. Braun, "Eigenvector error bounds and their applications to structural modifications," *AIAA Journal*, vol. 31, no. 4, pp. 759–764, 1993.

[17] Y.M. Ram, S.G. Braun, and J.J. Blech, "Structural modification in truncated systems by the Rayleigh-Ritz method," *Journal of Sound and Vibration*, vol. 125, no. 2, pp. 203–209, 1988.

[18] H.F. Weinberger, "Error bounds in the Rayleigh-Ritz approximation of eigenvectors," *Journal of Research of the National Bureau of Standards - B. Mathematics and Mathematical Physics*, vol. 64B, no. 4, pp. 217–225, 1960.

[19] O. C. Zienkiewicz, J. Bauer, K. Morgan, and E. Onate, "A simple and efficient element for axisymmetric shells," *International Journal for Numerical Methods in Engineering.*, vol. 11, pp. 1545–1558, 1977.

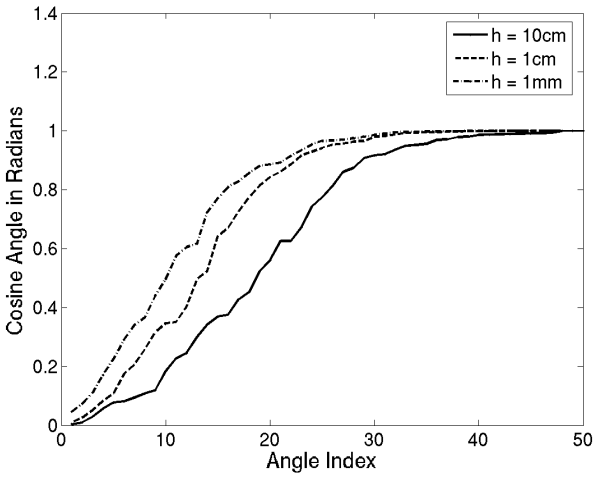


Figure 11: Cosine of angle between subspaces at two different steps.

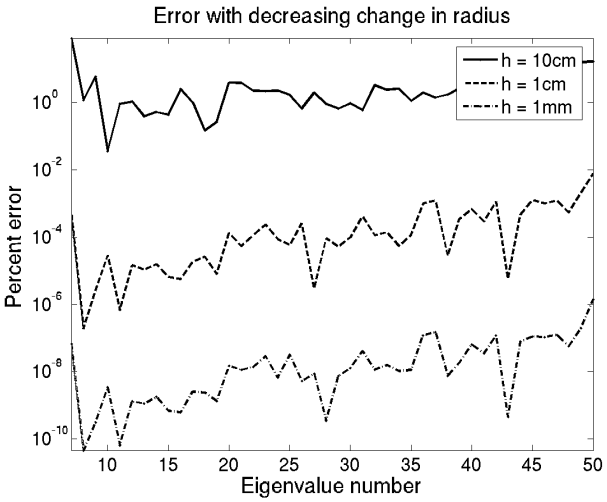


Figure 13: Error for different size h. 4 planes of symmetry.

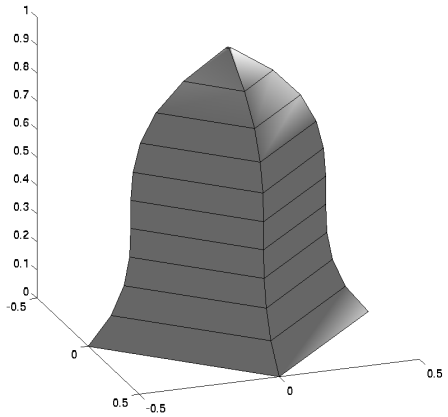


Figure 12: Whole bell geometry, 4 planes of symmetry.

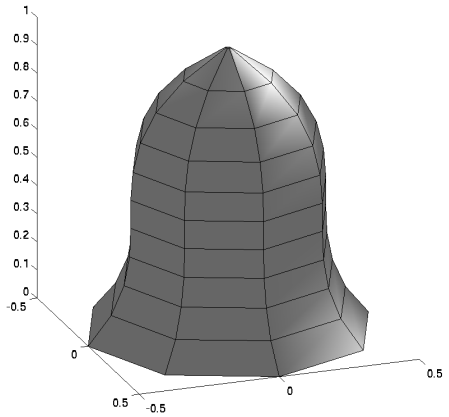


Figure 14: Whole bell geometry, 8 planes of symmetry.

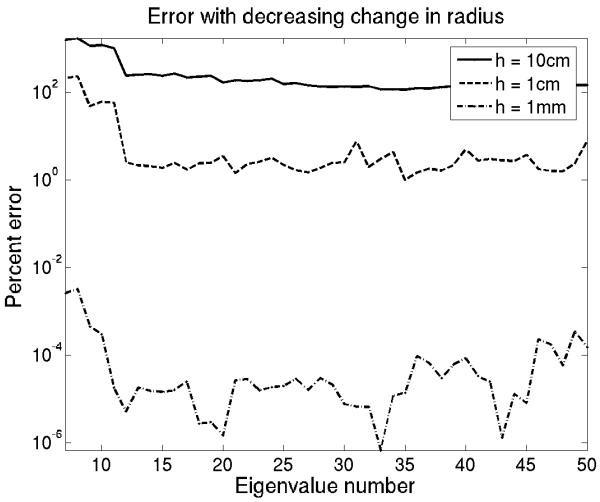


Figure 15: Error for different size h. 8 planes of symmetry.

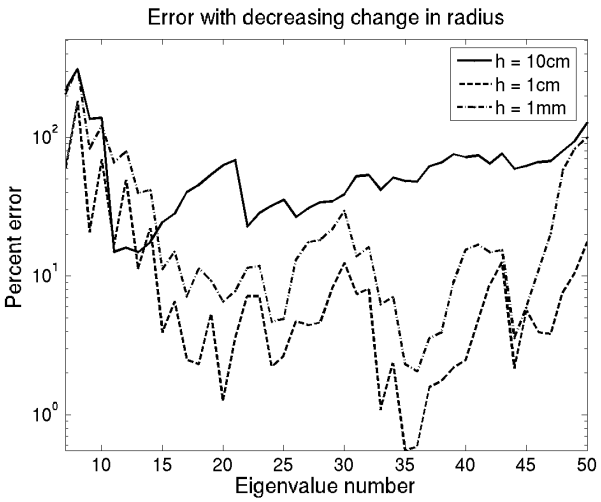


Figure 17: Error for different size h. 16 planes of symmetry.

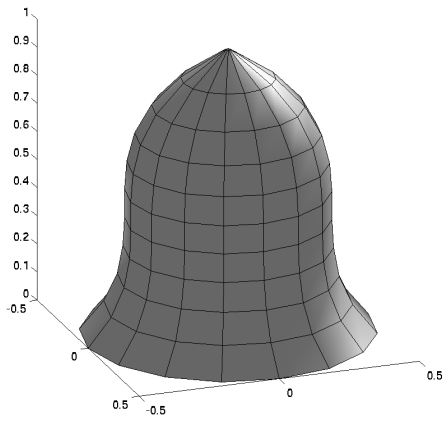


Figure 16: Whole bell geometry, 16 planes of symmetry.

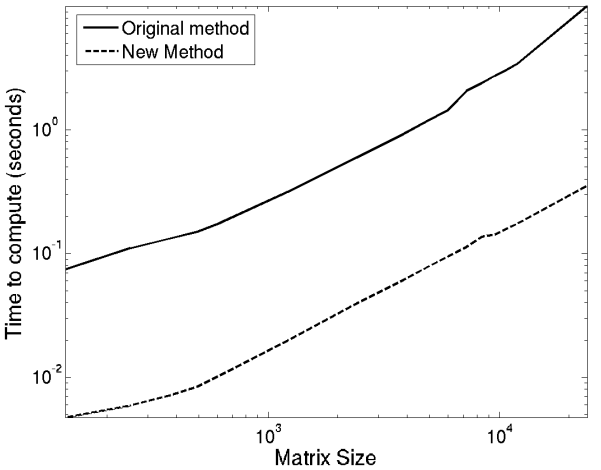


Figure 18: Time to compute new spectrum.

MUSICAL SIGNAL ANALYSIS USING FRACTIONAL-DELAY INVERSE COMB FILTERS

Vesa Välimäki, Heidi-Maria Lehtonen

Helsinki University of Technology, Laboratory of Acoustics and Audio Signal Processing
Espoo, Finland
vesa.valimaki@tkk.fi
heidi-maria.lehtonen@tkk.fi

Timo I. Laakso

National Board of Patents and Registration of Finland, Patents and Innovations Line
Helsinki, Finland
timo.laakso@prh.fi

ABSTRACT

A novel filter configuration for the analysis of harmonic musical signals is proposed. The method is based on inverse comb filtering that allows for the extraction of selected harmonic components or the background noise component between the harmonic spectral components. A highly accurate delay required in the inverse comb filter is implemented with a high-order allpass filter. The paper shows that the filter is easy to design, efficient to implement, and it enables accurate low-level feature analysis of musical tones. We describe several case studies to demonstrate the effectiveness of the proposed approach: isolating a single partial from a synthetic signal, analyzing the even-to-odd ratio of harmonics in a clarinet tone, and extracting the residual from a bowed string tone.

1. INTRODUCTION

Analysis of the amplitude envelope of harmonic components of a musical tone is a fundamental operation in musical signal processing. We discuss the harmonic extraction using the digital filtering approach. This is an old technique that has been proposed in different forms by Moorer in the 1970s for pitch detection of speech signals [1] and for analyzing music data for additive synthesis [2]. The basic idea is to use a multi-notch filter to extract individual harmonic components as signals. The filter structure may be obtained as the inverse transfer function of a comb filter (i.e., a delay line in a feedback loop).

In this paper we expand on a recently proposed idea that the delay line can be replaced with a high-order allpass fractional-delay filter to obtain very accurate cancellation of neighboring harmonics to extract a single harmonic [3]. The proposed signal analysis method is useful for many practical cases. Numerous musical instruments, including all woodwind, brass, and bowed string instruments, produce a sound signal that is inherently harmonic, i.e., the spectral components are integral multiples of a fundamental frequency. This follows from the sound-production mechanism of these self-excited systems, which involves mode locking in the time domain [4]. It forces the sustained tones of such instruments to be periodic. There is often a noise component in these musical tones making them pseudo-periodic in practice.

Another method for this kind of signal decomposition is sinusoidal modeling [5], [6], [7]. In this method the signal is analyzed using the windowed FFT, and the frequency and amplitude tracks are obtained by connecting data in the neighboring analysis frames. This approach has its roots in the phase vocoder technique and its efficient transform-domain implementation. For periodic or pseudo-periodic musical tones it is unnecessary to get down to an overly generic analysis method, because the frequencies of the harmonic components are known after the estimation of the fundamental frequency. Advantages of the proposed filter-based analysis method – compared with the more general FFT-based techniques – are simplicity, which follows mainly from the small number of parameters, and the possibility of designing filter coefficients in closed form. Additionally, the resulting decomposition is obtained directly as a set of time-domain signals, and no separate synthesis stage is required.

Other signal processing methods proposed for analyzing the harmonic structure of musical signals include wavelets [8] and high-resolution tracking methods [9], [10]. These methods provide excellent frequency accuracy at the expense of a complicated algorithm and a high computational cost. The method proposed in this paper can also provide amplitude and frequency accuracy that is sufficient for musical signal analysis but at the same time the analysis method remains easy to apply.

This paper is organized as follows. Section 2 discusses the filter structure for canceling harmonics of a musical signal, and Section 3 introduces a filter structure for extracting a single harmonic component and another structure for separating even and odd harmonics. In Section 4, three test cases are presented to demonstrate the power of this approach in musical signal analysis.

2. FRACTIONAL-DELAY INVERSE COMB FILTERS

The inverse comb filter[1] (ICF) is an FIR filter where the input signal is delayed by L samples and is then subtracted from the original input signal, see Fig. 1(a). The corresponding transfer function is $H(z) = (1 - z^{-L})/2$, where the scaling factor ½ sets the gain to unity in the passband (i.e., between the notches). The magnitude response of this filter features periodic notches at the multiples of f_s/L, where f_s is the sampling rate (Hz) and L is the delay line length in samples, or multiples of the sampling interval.

When the delay line length is restricted to be an integral multiple of the sample interval, the accuracy of the notch frequencies can be poor. An example is shown in Fig. 2 where the fundamental frequency is 4186 Hz and the corresponding period length is 10.5351 samples. Practical ICF implementations employ a fractional-delay filter that replaces the delay line [11], [13], [14]. Alternatively, an FIR [15], [16] or an IIR notch filter [17], [18] can be designed to approximate the overall ICF characteristics.

Figure 1(b) shows the block diagram of a fractional-delay ICF, where the delay line is replaced with an allpass filter, as pro-

[1] Following the convention of [11], the term 'inverse comb filter' is used for the feedforward system with a delay line. The 'comb filter' has a delay line inside a feedback loop.

posed previously [3]. The transfer function of this system can be written as $H_{fd}(z) = [1 - A(z)]/2$, where $A(z)$ is the transfer function of the allpass filter used for delay approximation. A magnitude response of this structure with an 11th-order allpass filter that approximates the delay of 10.5351 sampling intervals is displayed in Fig. 2 (solid line).

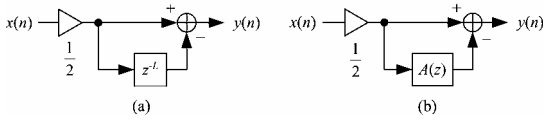


Figure 1: *(a) Conventional ICF and (b) a fractional-delay allpass-filter based ICF (after [3]).*

The transfer function of a digital allpass filter is

$$A(z) = \frac{z^{-N} D(z^{-1})}{D(z)} \tag{1}$$

where N is the order of the filter and $D(z) = 1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_N z^{-N}$ is the denominator polynomial with real-valued coefficients a_k, and the numerator polynomial is a reversed version of the denominator. The symmetry of the numerator and denominator coefficients guarantees the exact allpass property even for rounded coefficients. In this application, the allpass filter order is typically N = round(L), which is also approximately the period length (in samples) to be cancelled. Therefore, the filter order N can be very high, such as $N = 1000$ for a low fundamental frequency of 44.1 Hz when the sampling rate is 44.1 kHz. Evidently, a method is needed that allows for the design of high-order filters.

We propose two new structures, which are presented in Fig. 3. These filter structures offer freedom in the selection of the allpass filter order, which was related to the fundamental period in a previous work [3]. We have found experimentally that the order of $A(z)$ may be kept constant (e.g., $N = 80$), so when the fundamental period ($T_0 = f_s/f_0$) is longer than N samples, L extra samples of delay are required in the lower signal path in Fig. 3(a). However, when the fundamental period is shorter than N samples, K extra samples are required in the upper signal path to synchronize signals for subtraction, see Fig. 3(b). Thus, we propose to use the transfer function

$$H_{\text{low}}(z) = \frac{1}{2}\left[1 - z^{-L} A(z)\right] \tag{2}$$

when the fundamental period T_0 is larger than (or about the same as) the allpass filter order N, and the transfer function

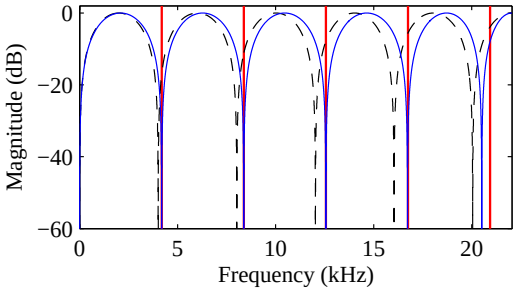


Figure 2: *Magnitude response of the conventional (dashed line) and the allpass-based (solid line) ICF. The thick vertical lines indicate the harmonic frequencies to be cancelled (f_0 = 4186 Hz).*

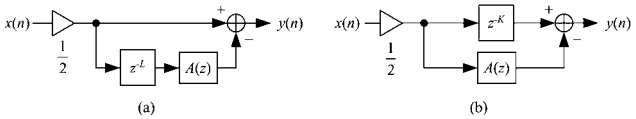


Figure 3: *Fractional-delay ICF structures that allow the use of an allpass filter A(z) of arbitrary order for signals with both (a) low and (b) high fundamental frequency.*

$$H_{\text{high}}(z) = \frac{1}{2}\left[z^{-K} - A(z)\right] \tag{3}$$

when the fundamental period T_0 is smaller than the allpass filter order N. The delay-line lengths L and K are determined as follows:

$$L = T_0 - N - d \text{ , when } T_0 \geq N \tag{4}$$

$$K = N + d - T_0 \text{ , when } T_0 < N \tag{5}$$

where $-1 < d < 1$ is the fractional-delay parameter used in designing the allpass filter.

2.1. Properties of allpass fractional-delay inverse comb filters

Let us consider the properties of the allpass-based ICF structure of Fig. 3. For simplicity, we will consider the case where $K = L = 0$, i.e., no additional delay is present in either branch, as in Fig. 1(b). We can express the transfer function (2) in the form

$$H(z) = \frac{1}{2}\left[1 - A(z)\right] = \frac{1}{2}\left[\frac{D(z) - z^{-N} D(z^{-1})}{D(z)}\right] \tag{6}$$

where the numerator polynomial can be written as

$$B(z) = D(z) - z^{-N} D(z^{-1}) = b_0 + b_1 z^{-1} + \ldots + b_N z^{-N} \tag{7}$$

and

$$b_k = a_k - a_{N-k}, \quad k = 0,1,\ldots,N \text{ .} \tag{8}$$

It is easy to verify that $B(z)$ is an *antisymmetric* polynomial, i.e., $b_k = -b_{N-k}$. In fact, the ICF is a special case of the parallel connection of two allpass filters discussed by Saramäki in [18]. As shown in this seminal paper, under general assumptions (stable allpass functions) the overall numerator polynomial of this structure is either symmetric or antisymmetric (mirror-image or anti-mirror-image polynomial, respectively). These polynomials are known to have their zeros exactly on the unit circle. Hence, our filter is known to have accurate zeros also in the fractional-delay ICF case. Note that the allpass filters are exactly allpass (unity magnitude at all frequencies) even if the phase (or phase delay, or group delay) is only approximately as desired.

The zeros of the conventional ICF (integer delay $L = L_0$) are known to be uniformly distributed on the unit circle:

$$H(z) = \frac{1}{2}(1 - z^{-L}) = 0 \quad \Leftrightarrow \quad z_n = e^{jn2\pi/L}, \; n = 0,1,\ldots,L-1 \tag{9}$$

In addition, there are L poles at $z = 0$. The frequency response of the ICF is obtained in the form

$$H(e^{j\omega}) = j e^{-j\omega L/2} \sin(\omega L/2) \equiv H_0(e^{j\omega}) \tag{10}$$

which will be used as a reference when comparing against another ICF structure.

For the fractional-delay ICF ($L = L_0 + d$, d real-valued) the zeros are difficult to express in general, as they depend on the approximating allpass filter of (6). However, the following notation is useful:

$$H(z) = \frac{1}{2}\left[1 - A(z)\right] = \frac{1}{2}\left(1 - z^{-L}\right) + \frac{1}{2}\left[z^{-L} - A(z)\right] \qquad (11)$$

where the latter term represents the error due to the allpass filter approximation. Note that in the z-transform formulation (11), the term z^{-L} with a noninteger power is non-realizable. A more practical expression is the frequency-domain for

$$\begin{aligned} H(e^{j\omega}) &= je^{-j\omega L/2}\sin(\omega L/2) + \frac{1}{2}\left[e^{-j\omega L} - A(e^{j\omega})\right] \\ &= H_0(e^{j\omega}) + \frac{1}{2}\left[e^{-j\omega L} - e^{j\phi_A(\omega)}\right] \end{aligned} \qquad (12)$$

The latter term utilizes the exact unit magnitude property of the allpass functions, which enables the expression with the corresponding phase function only. Hence, (12) illustrates the error term in the fractional-delay ICF frequency response caused by allpass phase approximation and the deviation from the ideal linear phase. As the phase approximation errors of the allpass filter tend to accumulate with increasing frequency, also the zeros of the corresponding ICF are more off the ideal places at higher frequencies.

2.2. Allpass fractional-delay filter design

Three closed-form design methods are known for fractional-delay allpass filters: the Thiran allpass filter design [20], [19], [11], the truncated Thiran allpass filter [21], and the Pei-Wang method [22]. Such methods are needed to increase the allpass filter order to be large enough for good wideband approximation in audio applications. Both the standard and the truncated Thiran methods allow the filter order to be increased up to $N = 1029$ (when $d = -0.5$) using 64-bit double floating-point computing.

The Thiran design formula can be expressed as

$$a_k = (-1)^k \binom{N}{k} \prod_{n=0}^{N} \frac{d+n}{d+k+n} \quad \text{for } k = 1, 2, 3, ..., N, \qquad (13)$$

where N is the filter order and d is the fractional delay parameter ($-0.5 < d \leq 0.5$). At low frequencies, this filter has the phase delay of $N + d$ samples. This design method was used to produce Fig. 2 and Fig. 4(b) with parameter values $N = 11$ and $d = -0.4649$.

The truncated Thiran design is obtained by modifying (13):

$$a_k = (-1)^k \binom{M}{k} \prod_{n=0}^{M} \frac{d+n}{d+k+n} \quad \text{for } k = 1, 2, 3, ..., N, \qquad (14)$$

where M is the prototype filter order ($M > N$) [21]. By using a value for M that is much larger than N in (14), it is possible to extend the bandwidth of good approximation. This comes at the expense of losing quality at low frequencies: the approximation error is larger than in the original allpass filter. This design technique allows a useful tradeoff between approximation accuracy and bandwidth, as discussed in [21] and [3].

Figure 4 compares the standard and truncated Thiran allpass filters. The design parameters are $N = 80$ and $d = -0.5$ for both filters, and the prototype filter order for the truncated Thiran filter is $M = 9N = 720$. The magnitude response, which is exactly flat in both cases, the phase delay (i.e., the negative phase function divided by angular frequency), and the frequency-response error (i.e., difference between frequency responses of the allpass filter and the ideal fractional delay element $e^{-j\omega D}$) are displayed. It is seen that the difference between the fractional-delay approximation of the two filters is microscopic below about 17 kHz, see Fig. 4(b), but the relaxed accuracy allows for the truncated Thiran filter to perform significantly better above 17 kHz, see Fig. 4(c).

A comparison of two ICFs based on allpass filters is shown in Fig. 5. The same design parameters were used as in Fig. 4, and the delay-line lengths were chosen to be $L = K = 0$. The impulse responses of the two ICFs are very similar, but not identical (see Fig. 5(a)). The magnitude responses in Fig. 5(b) are also nearly identical except at frequencies close to the Nyquist limit. In Fig. 5(c) it is seen that below 17 kHz the ICF using the truncated allpass filter is worse than the ICF using the standard Thiran filter, but both are sufficiently good, because the attenuation is more than 140 dB. Above 17 kHz the performance of the Thiran ICF collapses, but with the truncated version of the allpass filter the ICF offers an attenuation of 140 dB up to 20 kHz.

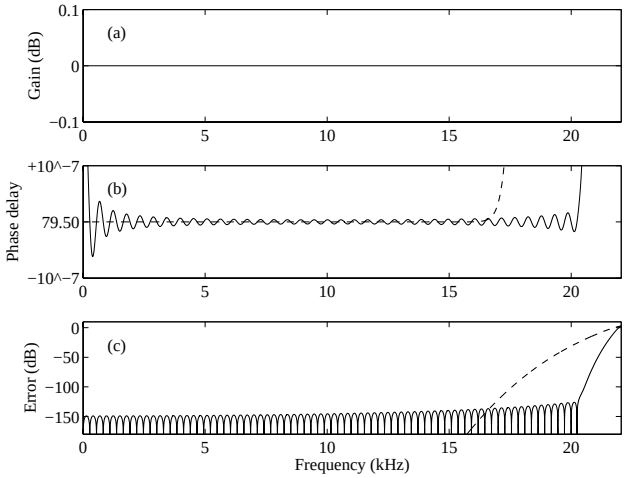


Figure 4: *(a) Magnitude response, (b) phase delay (in samples), and (c) frequency-response error of the Thiran allpass filter (dashed line) and the truncated Thiran allpass filter (solid line). The parameter values are N = 80, M = 720, and d = –0.5.*

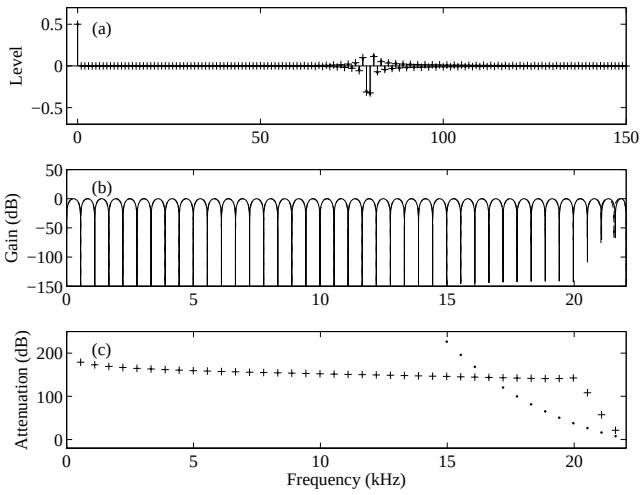


Figure 5: *(a) Impulse responses, (b) magnitude responses, and (c) harmonic attenuation of ICFs with the Thiran ('.', dashed line in (b)) and truncated Thiran allpass filters ('+', solid line in (b)).*

3. EXTRACTING HARMONIC COMPONENTS

Instead of canceling all the harmonic components, single harmonics can be extracted. This is achieved by cascading with an ICF a second-order all-pole filter that cancels a zero at a given harmonic frequency. This section describes the design of such a filter, which we call the *harmonic extraction filter* (HEF).

3.1. Harmonic extraction filter

It is not recommended to place a pole exactly on the unit circle in the z plane, because the resulting second-order filter is marginally stable and the hidden pole may cause numerical problems. A better approach is to move the zeros of the ICF slightly inside the unit circle by defining the radius of all zeros of the transfer function to be $r = 1 - \varepsilon$, where ε is a very small non-negative constant. Consequently, the pole of the second-order filter can also have the same radius, so that the stability of the recursive filter can be assured.

To place all the zeros at radius r, the coefficient of an ICF with a delay-line of length L must have a filter coefficient r^L [23]. Consequently, a scaling coefficient

$$g_0 = 1/(1+r^L) \tag{15}$$

must be used to ensure the maximum gain of the filter to be unity (i.e., 0 dB). Then, the minimum gain of the filter, which occurs at the bottom of the notches at harmonic frequencies, is $g_0(1 - r^L)$, which we call A. We can now solve for the required g_0, and consequently the required r, when the gain A is set to a given value. From

$$A = g_0(1-r^L) = (1-r^L)/(1+r^L) \tag{16}$$

it follows that

$$r^L = (1-A)/(1+A)\,. \tag{17}$$

Since the radius of all zeros is r, the pole radius must also be selected to be r. Based on (17) the radius can be determined to be

$$r = \sqrt[L]{(1-A)/(1+A)}\,. \tag{18}$$

The HEF transfer function can be written as

$$H_{\mathrm{HEF}}(z) = g_1 R(z)[1 - r^L A(z)] \tag{19}$$

where the scaling coefficient g_1 that sets the maximum gain at the bottom of the notches (without the resonator) to be unity is

$$g_1 = 1/\left[r(1-r^L)\right] \tag{20}$$

and the transfer function of the resonant filter is

$$R(z) = b_0/(1 + a_1 z^{-1} + a_2 z^{-2}) \tag{21}$$

with coefficients $b_0 = (1 - r^2)\sin(2\pi f_{\mathrm{res}}/f_{\mathrm{s}})$, which scales the maximum gain of the resonant filter to be unity (see, e.g., [11]), $a_1 = -2r\cos(2\pi f_{\mathrm{res}}/f_{\mathrm{s}})$, $a_2 = r^2$, and f_{res} is the resonance frequency that determines which harmonic component is retained. The filter that has the transfer function (19) with the given scaling coefficients has a maximum gain of 0 dB at the peak of the passband.

When the delay-line length is an integer, the allpass filter $A(z)$ is reduced to a delay line. The zeros of this integer-delay HEF are inside the unit circle, on a smaller circle with radius r:

$$\begin{aligned} &H_{\mathrm{HEF}}(z) = g_1 R(z)\left[1 - r^L z^{-L}\right] = 0 \\ &\Leftrightarrow \quad z_n = r e^{jn2\pi/L},\ n = 0,1,\ldots,L-1 \end{aligned} \tag{22}$$

The transfer function can be given in the form

$$H_{\mathrm{HEF}}(z) = g_1 R(z)\left[r^L(1-z^{-L}) + (1-r^L)\right] \tag{23}$$

which enables the frequency-domain expression

$$H_{\mathrm{HEF}}(e^{j\omega}) = g_1 R(e^{j\omega})\left[2r^L H_0(e^{j\omega}) + (1-r^L)\right] \tag{24}$$

This illustrates the constant (frequency-independent) term due to the zeros being placed inside the unit circle. Since all the zeros are inside the unit circle, the overall magnitude never reaches zero exactly.

Finally, we obtain the expressions

$$H_{\mathrm{HEF}}(z) = g_1 R(z)\left[r^L(1-z^{-L}) + (1-r^L) + r^L(z^{-L} - A(z))\right] \tag{25}$$

and

$$\begin{aligned} &H_{\mathrm{HEF}}(e^{j\omega}) \\ &= g_1 R(e^{j\omega})\left[2r^L H_0(e^{j\omega}) + (1-r^L) + r^L(e^{-j\omega L} - e^{j\phi_A(\omega)})\right] \end{aligned} \tag{26}$$

where both the constant and allpass phase dependent errors in the fractional-delay HEF response are visible.

3.2. Design of parameter values

For good attenuation, it is required that A is sufficiently small and that the resonant filter $R(z)$ accurately cancels one of the zeros of the ICF. For example, when it is required that the inverse comb filter attenuates the harmonic frequencies by 100 dB, the value of A must be set to 10^{-5}, since $20\log_{10}(10^{-5}) = -100$ dB.

In practice, the high-order allpass filter does not provide a perfect phase approximation. Thus, it may be necessary to set A to a smaller value, such as 10^{-6}. However, when the filter structure for selecting a single harmonic is used, the resonant filter provides additional attenuation at frequencies away from the resonance, which further improves the attenuation at the notches.

It was reported in [3] that for some musical instrument tones with strong low-indexed harmonics, the filtering of the signal with the transfer function $H_{\mathrm{HEF}}(z)$ is insufficient. Listening to the filtered signal reveals that the fundamental frequency is still perceived although one of the high-frequency partials is strongly emphasized. Filtering the signal twice with transfer function (19) adequately attenuates the rest of the harmonics in this case.

There is a minor mismatch in the cancellation of the mth transfer function zero with the pole of the resonant filter with the resonance frequency $f_{\mathrm{res}} = mf_0$, because the frequency of the mth zero is offset by the inaccuracy of the phase approximation of the allpass filter. In practice, this mismatch produces a kink around the main lobe of the bandpass filter, and the gain at the resonance frequency becomes larger than 0 dB. A correction to the pole frequency is required to reduce this error.

One way to correct the resonance frequency of the all-pole filter is to search for the minimum of the ICF's magnitude response around the mth notch. For example, computing the magnitude response at 10,000 points between $0.999990mf_0$ and $1.000010mf_0$, and selecting f_{res} as the frequency, where the minimum occurs, reduces the mismatch sufficiently. To reduce the number of magnitude response evaluations, the local minimum can be estimated by using interpolation. After a local minimum on a coarse grid of spectral points has been found, two straight lines can be fit through it and its neighboring points. It may be assumed that the slopes of the notch are symmetrical, so that the frequency of the actual minimum can be found where the two lines cross. This method gives accurate enough results at low computational costs, and considerably improves the performance of the algorithm.

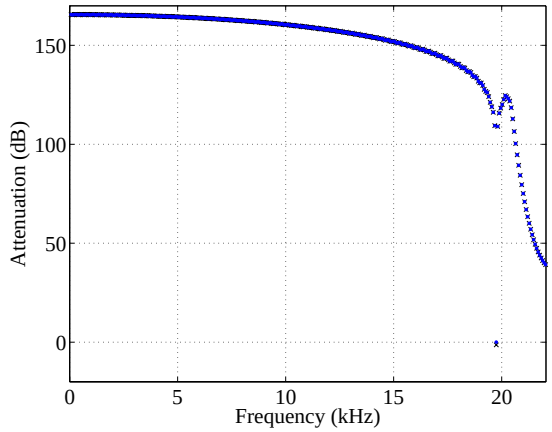


Figure 6: *Attenuation of harmonic partials using the single-harmonic canceling filter when the resonance frequency is the nominal mf_0 ('x'), and the corrected one ('.'). Notice that the largest difference between these data occurs at the frequency of the harmonic #285 at 19.7 kHz.*

Fig. 6 gives an example of the attenuation obtained without and with the proposed correction of the resonance frequency when the fundamental frequency is 69.2957 Hz, the harmonic #285 at 19749.2 Hz is selected, the allpass filter orders used are $N = 80$ and $M = 720$, and attenuation is $A = 10^{-5}$. In this case, the pole radius is $r = 1 - 31.4 \times 10^{-9} = 0.999999969$. The difference between the nominal (mf_0) and the corrected resonance frequency is 13.5×10^{-3} Hz, but the attenuation of the partial is 1.4 dB without and 0.0051 dB with the correction. This difference is enough to make the correction worth the effort, since it makes the analysis filter an accurate tool for signal analysis.

For comparison, we designed a linear-phase FIR bandpass filter that imitates the obtained magnitude response. The filter was designed by using the Chebyshev window with a sidelobe level of –100 dB. To extract the harmonic #285 and to obtain an attenuation of more than 100 dB for all other harmonics, the smallest filter order is 4657. The proposed allpass-filter based ICF of order 80 is computationally much less expensive. Its number of filter coefficient is 1.7% of that of the FIR filter.

3.3. Separation of odd and even partials

While it is possible to cancel the harmonic components one by one by applying the above HEF structure multiple times, alternatively even and odd harmonics may be separated using a single filtering operation, as suggested in [3]. The odd and even harmonics can be separated by first canceling the even harmonics using the fractional-delay inverse comb filter and then subtracting the resulting signal from the original.

The structure of Fig. 3 can be used, but the delay to be approximated is half of that used in canceling all harmonics, i.e., $f_s/2f_0$ samples. With this delay, the notches are located at the multiples of the second harmonic, and the filter now cancels the even harmonics and preserves the odd harmonics. The signal containing even harmonics is then obtained by subtracting the estimated odd harmonics from the original signal, as shown in Fig. 7:

$$s_{\text{even}}(n) = s_{\text{orig}}(n) - s_{\text{odd}}(n). \tag{27}$$

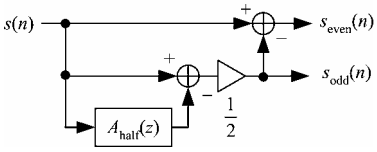


Figure 7: *Structure for separating the even and odd harmonics of a musical signal using one allpass filter.*

4. CASE STUDIES

This section presents how the proposed filtering algorithms perform with synthetic signals and recorded instrument tones. In addition, the proposed filter is compared against two techniques: a fractional-delay FIR filter using the well known Lagrange interpolation [11] and sinusoidal modeling [5], [6], [7]. The sampling frequency is 44.1 kHz in all the test cases.

4.1. Harmonic extraction from a synthetic test signal

The following example illustrates how the algorithm works with a synthetic test signal. The signal is determined to be the sum of sinusoids:

$$x(n) = A_{\text{sc}}(n)\sum_{k=1}^{K}\sin\left(\frac{2\pi nkf_0}{f_s} + \varphi_k\right), \tag{28}$$

where $A_{\text{sc}}(n)$ is an envelope function, K is the number of harmonics present in the signal, f_0 is the fundamental frequency of the signal, and φ_k is the phase of the kth harmonic. In this case, the parameters were chosen as follows: $K = 84$, $f_0 = 261.626$ Hz (C4), which corresponds to the cycle length of 168.562 samples. The initial phases φ_k are uniformly distributed random numbers in the range $[0, 2\pi]$. In order to examine the temporal smearing resulting from the harmonic component extraction, the envelope of the signal is chosen to be rectangular, containing sharp transitions.

As an example, two components, the fundamental frequency component and the 76th overtone, have been extracted from the signal (28). This is done with the HEF structure (19). The order of the truncated Thiran filter was chosen to be $N = 80$ and the order of the prototype filter was set to $M = 9N$. The attenuation coefficient A was determined to be equal to 10^{-6}. A comparison against a Lagrange FIR filter and sinusoidal modeling technique was carried out. The order of the Lagrange FIR filter N_{L} was set to 80 so that the amount of coefficients is the same as that of the proposed filter. In sinusoidal modeling, the following parameters were used. The short-time FFT was computed using a Blackman window of length $4f_s/f_0$, and the FFT size was 2048. The hop size was set to be one-fourth of the window length.

The results are presented in Figs. 8 and 9. Figs. 8(a) and (b) present the spectrum of the original signal (28), and Figs. 8(c) and (d) present the spectrum of the partials #1 and #76 that are extracted with the proposed method. Figs. 8(e) and (f) present the corresponding result obtained with the Lagrange FIR filter, and Figs. 8(g) and (h) represent the result obtained with sinusoidal modeling. All spectra were calculated from a 0.5 s excerpt taken between 0.1 s and 0.6 s of the signals. The Hamming window was used, and the spectra were computed using the discrete-time Fourier transform at 851 equally spaced points so that every 10th point matched one harmonic. This choice of parameters yields a clear visual representation of the sharp spectral peaks.

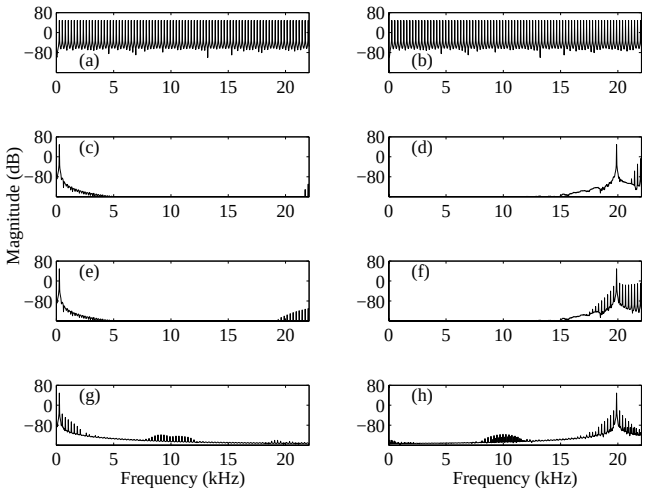


Figure 8: *Results of comparison in the frequency domain. (a), (b) Synthetic test signal and harmonic components #1 and #76 obtained with (c), (d) with the proposed method, (e), (f) with Lagrange FIR filter, and (g), (h), with sinusoidal modeling.*

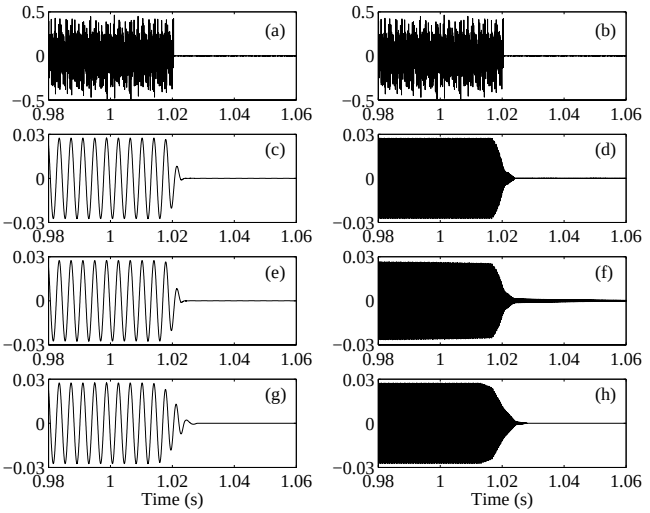


Figure 9: *The effects of temporal smearing. (a) and (b), an excerpt of the original synthetic test signal, (c) and (d) excerpts of the 1st and 76th harmonic obtained with the proposed allpass filter, (e) and (f) with the Lagrange FIR filter, and (g) and (h), with sinusoidal modeling.*

As can be seen in Figs. 8(c) and (e), the proposed filter and Lagrange filter are able to extract the first harmonic efficiently, and the other harmonics are properly attenuated. In the case of the 76th harmonic component, some of the uppermost harmonics are not properly attenuated because of the error in the phase delay near the Nyquist limit (see Fig. 4) and its effect on the attenuation of the inverse comb filter. The Lagrange FIR filter in Figs. 8(e) and (f), the error near the Nyquist limit is greater, and moreover, the lowpass nature of the magnitude response has to be taken into account in the analysis in order to maintain the level of the original signal at high frequencies. The sinusoidal modeling technique depicted in Figs. 8(g) and (h), suffers from sidelobes of the window function, and the attenuation of the neighboring harmonics is not as efficient as with the other methods.

The effects of temporal smearing are illustrated in Fig. 9. The signals are zoomed to the window 0.98 – 1.06 s. Figs. 9(a) and (b) show the original signal and Figs. 9(c) and (d) present the cases where the first harmonic and the 76th harmonic have been extracted with the proposed method, respectively. The performance of the Lagrange FIR filter is depicted in Figs. 9(e) and (f). It can be seen in Fig. 9(e) that in the case of the first harmonic the effect of temporal smearing is about same as with the proposed method. However, the smearing is greater in higher frequencies, which is visible in Fig. 9(f), since the lowpass nature of the filter complicates the usage of the resonator. In this case, the magnitudes of the resonator and the Lagrange filter do not compensate each other, which leads to improper attenuation. The temporal smearing in sinusoidal modeling technique depicted in Figs. 9(g) and (h) is slightly greater than that of the other methods.

4.2. Even-to-odd ratio calculation

In the case of the clarinet, the relation of even and odd harmonics and its effect to the timbre has been studied by Barthet *et al.* [27]. They have derived new descriptors for the clarinet timbre by constructing a simple but efficient parametric model for the clarinet to control certain parameters: the dimensionless mouth pressure γ and the embouchure parameter ζ, in addition to the fundamental frequency of the bore f_b and the reed resonance frequency f_r. The γ parameter defines the ratio between the pressure inside the player's mouth and the static beating reed pressure. The ζ parameter, in turn, takes the lip position and the section between the mouthpiece opening and the resonator into account. By varying these parameters Barthet *et al.* derived a relation between the parameters and the timbre characteristics.

We have examined the synthetic sounds generated by Barthet et al. [27] with the proposed algorithm in the case where the parameter γ varies between 0.40 and 0.50. As the relation of the odd and even harmonics has an effect on the spectum and timbre of the sound, the odd and even harmonics were separated with the structure presented in Fig. 7. The original signal and the separation results are shown in Fig. 10 for γ = 0.40. The other parameters, ζ, f_b,and f_r, are equal to 0.33, 170 Hz and 2500 Hz, respectively. The magnitude response is calculated with a 512-point FFT from an excerpt taken from 0.5 – 1.0 s. A Hamming window is used in the computation.

As Fig. 10 shows, the magnitude of the odd harmonics is greater than that of the even harmonics, which is typical for the clarinet sound. The odd-to-even ratio is determined by first calculating the envelopes of both signals. This is done by averaging the full-wave rectified signal over a window of 500 samples. The ratio between the magnitude of the odd and even harmonics is obtained by dividing the envelopes. The result seems to depend on the parameter γ. This relation is illustrated in Fig. 11 for four different values of γ (0.40, 0.42, 0.47, and 0.50).

Fig. 11 shows that the difference between the player's mouth pressure and the static beating reed pressure affects the odd-to-even ratio. That is, with larger pressure differences, the proportion of the odd harmonics in the tone is greater than in the case when the pressure difference is smaller (smaller γ value). Also the static state is reached faster with a larger pressure difference.

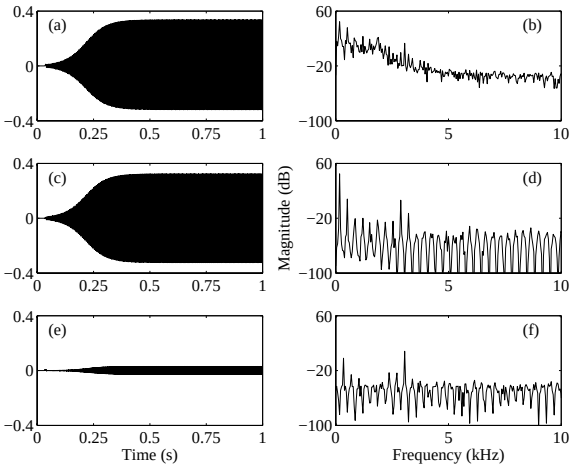


Figure 10: *(a) and (b) The synthetic clarinet tone in time and frequency domains, respectively, (c) and (d) the separated odd harmonics in time and frequency domains, respectively, and (e) and (f) even harmonics in the time and frequency domains ,respectively.*

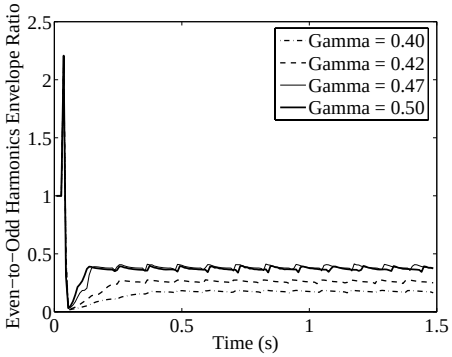


Figure 11: *The ratio of even and odd harmonics with four different γ values.*

4.3. Residual signal extraction

In order to investigate how the proposed algorithm works with recorded tones, a double bass tone was analyzed with the proposed method. In Figs. 12(a) and (b), the tone is presented in the time and frequency domains, respectively. The fundamental frequency of the tone was measured to be f_0 = 58.2670 Hz.

It is now desired to remove all the harmonics instead of preserving one. A modified form of the transfer functions (2) or (3) can be used, depending on the fundamental frequency,

$$H_{\mathrm{HEF}}(z) = g_1\left[1 - r^L A(z)\right], \tag{29}$$

where the coefficients g_1 and r^L are determined in the same manner as described in Sec. 3.1. $A(z)$ written is as in (1). The order of the truncated Thiran filter is N = 80 and the order of the prototype filter is $M = 9N$. The attenuation coefficient A is set to 10^{-6}. The filtered residual signal is presented in Figs. 12(c) and (d) in time- and frequency domains, respectively. When comparing Figs. 12(b) and (d), it is seen that the harmonic components are attenuated efficiently. Moreover, the noise between the harmonics is preserved.

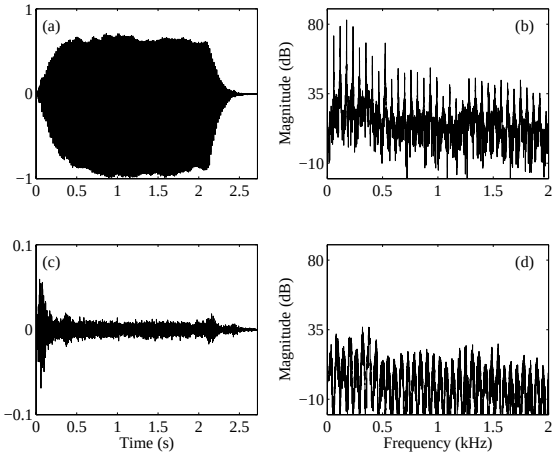


Figure 12: *Time- and frequency-domain presentations of (a), (b) the double bass tone and (c), (d) the extracted residual signal.*

5. CONCLUSIONS AND FUTURE WORK

Digital filtering techniques were proposed to obtain useful decompositions of harmonic musical signals. The basic approach taken here is to subtract a delayed copy of the signal from itself to cancel the harmonic components. A high-order digital allpass filter implements an accurate approximation of the required time delay. A harmonic extraction filter is obtained by cascading a second-order all-pole filter with the inverse comb filter. Division of a musical signal into two signals, one containing the even and the other the odd harmonics, and the extraction of the background noise or residual were also suggested as promising operations that are easy to realize using the proposed filter structures.

Case studies were presented to show how the techniques perform in the feature analysis for musical tones. Single harmonic components were extracted from a synthetic test tone. The neighboring harmonics were attenuated more than 100 dB. The harmonic even-to-odd ratio was determined for synthetic clarinet tones. Finally, the residual noise component was extracted from a bowed string tone by suppressing all the harmonics.

Future research includes developing a useful method to account for varying fundamental frequency of the signal, such as vibrato. In practice, this problem calls for a time-varying delay line to be used in the inverse comb filter. There are known methods for modulation of the delay-line length for example in effects processing, such as flanging and chorus algorithms. Which interpolation technique should be used and for how fast and wide delay-length modulation can this method be accurate? A further application of the time-varying inverse comb filter would be the separation of harmonic audio signals, such as musical tones or voiced speech, as discussed by de Cheveigné [28].

Another interesting special case is the analysis of inharmonic tones, such as piano tones or other instrument sounds with regular inharmonicity caused by dispersion. A filter-based analysis tool for such tones requires an allpass filter that approximates the dispersion in cascade with the delay line. This is a known method in digital waveguide synthesis of string sounds, see, e.g., [29], [30]. A more accurate approximation of dispersion characteristics is needed for the analysis tool than for sound synthesis.

6. ACKNOWLEDGMENTS

The work of Heidi-Maria Lehtonen has been supported by Tekniikan edistämissäätiö, the Emil Aaltonen Foundation, the Finnish Cultural Foundation, and the GETA graduate school. The authors are grateful to Ms. Minna Ilmoniemi and Dr. Minna Huotilainen for their collaboration in the early stages of this work. Special thanks go to M. Barthet, P. Guillemain, R. Kronland-Martinet, and S. Ystad for allowing us to use their synthetic clarinet tones as test signals in this work.

7. REFERENCES

[1] J. A. Moorer, "The optimum comb method of pitch period analysis of continuous digitized speech," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. 22, no. 5, pp. 330-338, Oct. 1974.

[2] J. A. Moorer, "Signal processing aspects of computer music: a survey," *Proc. IEEE*, vol. 5, no, 8, pp. 1108-1137, 1977.

[3] V. Välimäki, M. Ilmoniemi, and M. Huotilainen, "Decomposition and modification of musical instrument sounds using a fractional delay allpass filter," in *Proc. Nordic Signal Processing Symp*., pp. 208–211, Espoo, Finland, June 2004. Available online at http://wooster.hut.fi/publications/nosig2004/60_VALIM.PDF.

[4] N. H. Fletcher and T. D. Rossing, *The Physics of Musical Instruments*. Springer, 1991.

[5] R. J. McAulay and T. F. Quatieri, "Speech analysis/synthesis based on a sinusoidal representation," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. 34, no. 4, pp. 744–754, 1986.

[6] X. Serra, *A System for Sound Analysis/Transformation/Synthesis Based on a Deterministic plus Stochastic Decomposition*. Ph.D. thesis. Report no. STAN-M-58, CCRMA, Stanford University, Stanford, CA, 1989. http://ccrma.stanford.edu/STANM/stanms/stanm58/.

[7] X. Serra and J. O. Smith, "Spectral modeling synthesis: A sound analysis/synthesis based on a deterministic plus stochastic decomposition," *Computer Music J*., vol. 14, no. 4, pp. 12-24, 1990.

[8] G. Evangelista, "Pitch synchronous wavelet representations of speech and music signals," *IEEE Trans. Signal Processing*, vol. 41, no. 12, pp. 3313–3330, Dec. 1993.

[9] B. David, R. Badeau, and G. Richard, "HRHATRAC algorithm for spectral line tracking of musical signals," in *Proc. Int. Conf. Acoustics, Speech, and Signal Processing*, Toulouse, France, May 15-19, 2006, vol. 3, pp. 45–48.

[10] R. Badeau, B. David, and G. Richard, "High resolution spectral analysis of mixtures of complex exponentials modulated by polynomials," *IEEE Trans. Signal Processing*, vol. 54, no. 4, pp. 1341–1350, Apr. 2006.

[11] K. Steiglitz, *A Digital Signal Processing Primer with Applications to Digital Audio*. New York: Addison Wesley, 1996.

[12] T. I. Laakso, V. Välimäki, M. Karjalainen, and U. Laine, "Splitting the unit delay—Tools for fractional delay filter design," *IEEE Signal Processing Mag.*, vol. 13, no. 1, pp. 30–60, Jan. 1996.

[13] V. Välimäki and T. I. Laakso, "Fractional delay filters—Design and applications," in *Nonuniform Sampling: Theory and Practice*. F. Marvasti, Ed. Kluwer Academic/Plenum Publishers: New York, 2001, Chapter 20, pp. 835–895.

[14] S. C. Pei and C.–C. Tseng, "A comb filter design using fractional-sample delay," *IEEE Trans. Circ. Syst.—Part II*, vol. 45, no. 6, pp. 649–653, June 1998.

[15] S. C. Dutta Roy, S. B. Jain, and B. Kumar, "Design of digital FIR notch filters," *IEE Proc. Vis. Image Signal Process*., vol. 141, no. 5, pp. 334–338, Oct. 1994.

[16] C.-C. Tseng and S.-C. Pei, "Sparse FIR notch filter design and its application," *Electronics Letters*, vol. 33, no. 13, pp. 1131–1133, June 1997.

[17] S.-C. Pei and C.-C. Tseng, "IIR multiple notch filter design based on allpass filter," *IEEE Trans. Circuits and Systems Part II: Analog and Digital Signal Processing*, vol. 44, no. 2, pp. 133–136, Feb. 1997.

[18] C.-C. Tseng and S.-C. Pei, "Stable IIR notch filter design with optimal pole placement," *IEEE Trans. Signal Processing*, vol. 49, no. 11, pp. 2673–2681, Nov. 2001.

[19] A. Fettweis, "A simple design of maximally flat delay digital filters," *IEEE Trans. Audio and Electroacoust*., vol. 20, no. 2, pp. 112–114, 1972.

[20] J.–P. Thiran, "Recursive digital filters with maximally flat group delay," *IEEE Trans. Circ. Theory*, vol. 18, no. 6, pp. 659–664, 1971.

[21] V. Välimäki, "Simple design of fractional delay allpass filters," in *Proc. European Signal Processing Conf.*, vol. 4, pp. 1881–1884, Tampere, Finland, Sept. 2000.

[22] S.-C. Pei and P.-H. Wang, "Closed-form design of all-pass fractional delay filters," *IEEE Signal Processing Letters*, vol. 11, no. 10, pp. 788–791, Oct. 2004.

[23] S. J. Orfanidis, *Introduction to Signal Processing*. Prentice Hall, 1996.

[24] T. Tolonen, "Methods for separation of harmonic sound sources using sinusoidal modeling," in *Proc. AES 106th Convention*, Preprint 4958, Munich, Germany, May 1999. http://lib.tkk.fi/Diss/2000/isbn9512251965/.

[25] P. A. A. Esquef, M. Karjalainen, and V. Välimäki, "Frequency-zooming ARMA modeling for analysis of noisy string instrument tones," *EURASIP J. Applied Signal Processing*, vol. 2003, no. 10, pp. 953–967, 2003.

[26] M. F. McKinney and J. Breebaart, "Features for audio and music classification", in *Proc. Int. Conf. Music Information Retrieval* (ISMIR), 2003. Available online at http://ismir2003.ismir.net/papers/McKinney.PDF.

[27] M. Barthet, P. Guillemain, R. Kronland-Martinet, and S. Ystad, "On the relative influence of even and odd harmonics in clarinet timbre," in *Proc. Int. Comp. Music Conf*., Barcelona, Spain, Sept. 2005.

[28] A. de Cheveigné, "Separation of concurrent harmonic sounds: Fundamental frequency estimation and a time-domain cancellation model of auditory processing," *J. Acoust. Soc. Am*., vol. 93, no. 6, pp. 3271–3290, June 1993.

[29] D. Rocchesso and F. Scalcon, "Accurate dispersion simulation for piano strings," in *Proc. Nordic Acoustical Meeting*, Helsinki, Finland, pp. 407–414, June 1996.

[30] J. Rauhala and V. Välimäki, "Tunable dispersion filter design for piano synthesis," *IEEE Signal Processing Letters*, vol. 13, no. 5, pp. 253–256, May 2006.

REAL-TIME AND EFFICIENT ALGORITHMS FOR FREQUENCY WARPING BASED ON LOCAL APPROXIMATIONS OF WARPING OPERATORS

Gianpaolo Evangelista
Sound Technology, Media Division
Inst. of Technology, Linköping University
Norrköping, Sweden
giaev@itn.liu.se

Sergio Cavaliere
ACEL
Dept. of Physical Sciences
Federico II University of Naples, Italy
sergio.cavaliere@na.infn.it

ABSTRACT

Frequency warping is a modifier that acts on sound signals by remapping the frequency axis. Thus, the spectral content of the original sound is displaced to other frequencies. At the same time, the phase relationship among the signal components is altered, nonlinearly with respect to frequency. While this effect is interesting and has several applications, including in the synthesis by physical models, its use has been so far limited by the lack of an accurate and flexible real-time algorithm. In this paper we present methods for frequency warping that are based on local approximations of the warping operators and allow for real-time implementation. Filter bank structures are derived that allow for efficient realization of the approximate technique. An analysis of the error is also presented, which shows that both numerical and perceptual errors are within acceptable limits. Furthermore, the approximate implementation allows for a larger variety of warping maps than that achieved by the classical (non-causal) first-order allpass cascade implementation.

1. INTRODUCTION

This paper is concerned with the computation of the frequency warping operation on signals, such as a musical tone of duration of about one second or longer. In other audio applications, frequency warping is often applied to filters and to filter design, in which case the computation presents little or no problem, the results are classical and well studied [1] and the technique is implementable in real-time. On the contrary, warping long-length signals presents large computational problems, both from the point of view of complexity and of causality, which hamper the possibility of using this technique in real-time applications, at least in exact form. A few plugins partially exploiting the musical capabilities offered by frequency warping are available on the market [2, 3].

In this paper, we present a simple approximation method that yields efficient and causal structures, in the form of unconventional multirate filter banks, which allow for real-time computation of frequency warping, for a large class of warping maps. At similar computational cost, the algorithm largely improves the quality of warping with respect to the one presented in [4, 5]. The new approximation is based on the intuitive idea that frequency warping a narrowband signal, such as a sinusoidal wave packet, simply amounts to properly time scale the amplitude envelope and to remodulate the signal to warped frequency. The scaling factor of the envelope approximately depends on the derivative of the warping map at warped frequency.

If a wideband signal is decomposed into sinusoidal wave packets, e.g., by means of a Gabor expansion or short-time Fourier transform [6, 7], then the warped signal can be generated by scaling, remodulating and properly delaying each packet. Due to the dispersive character of frequency warping, the original time shift of the wave packets transforms into a frequency dependent delay, which can be approximated to a constant delay for wave packets at each given frequency. Fortunately, scaling and delay transform in a covariant way by warping. As it can be expected, this results into generalized phase vocoder filter bank structures [8] where different resampling factors are applied to the various channels. Moreover, scaling is achieved by enforcing unequal downsampling and upsampling rates in each channel.

The efficient filter bank structures for the computation are discussed in this paper together with an analysis of the complexity and of the error. Our results show that both the numerical and the perceptual errors are largely tolerable.

The applications of the approximate warping filter bank structure range from signal representations [9] to synthesis and digital audio effects [10, 5]. In particular, in the physical models of piano strings and rods, the frequency warping section can be cascaded to a digital waveguide in order to simulate dispersion [11]. At comparable computational cost and similar warping map flexibility, this is a viable alternative to the use of dispersive waveguides that does not require the design of very high order allpass filters [12].

2. FREQUENCY WARPING OPERATORS

Frequency warping is completely characterized by a frequency transformation, i.e., by a map $\theta(\omega)$ of the frequency axis into or onto itself. Given a signal $s(t)$ with Fourier transform $S(\omega)$, the Fourier transform of the warped signal is the function

$$S(\theta(\omega)) = \tfrac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} s(t) e^{-j\theta(\omega)t} dt. \tag{1}$$

Thus, frequency warping can be defined as action of the operator

$$\mathfrak{W}_\theta = \mathfrak{F}^\dagger \mathfrak{C}_\theta \mathfrak{F}, \tag{2}$$

where $\mathfrak{F}$ is the Fourier transform operator and $\mathfrak{F}^\dagger$ its adjoint. The symbol $\mathfrak{C}_\theta$ denotes the composition by θ operator, i.e.,

$$[\mathfrak{C}_\theta S](\omega) = [S \circ \theta](\omega) = S(\theta(\omega)). \tag{3}$$

The arbitrary shape of the map θ defines the character of the frequency warping operations. In most applications, $\theta(\omega)$ is a continuously increasing and differentiable antisymmetric function mapping zero frequency to zero frequency, as shown in Figure 1. However, more general forms in which the warping map $\theta(\omega)$ is not

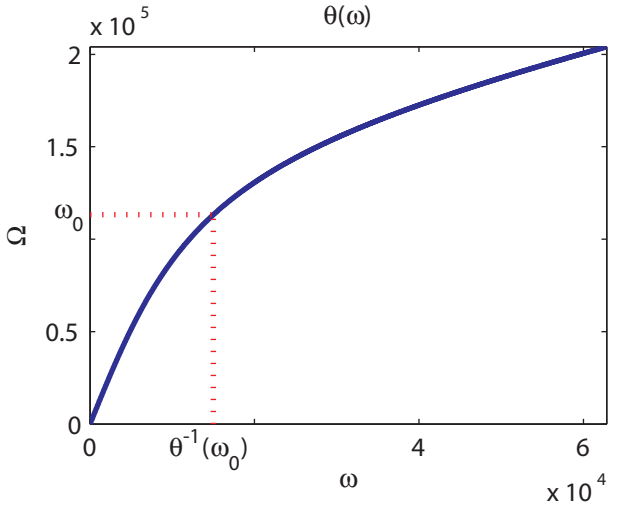


Figure 1: *A typical warping map.*

one-to-one can be considered provided that local invertibility is guaranteed.

If the map is one-to-one and almost everywhere differentiable then a unitary form of the warping operator can be defined by the action

$$\tilde{S}(\omega) = [\mathfrak{U}_\theta S](\omega) = \sqrt{\left|\frac{d\theta}{d\omega}\right|} S(\theta(\omega)). \tag{4}$$

We will assume henceforth that the map is increasing so that its first derivative is non-negative and the absolute value can be dropped in (4). The main property of the unitary warping operator is to preserve in-band energy. This is an important property even from the perceptual point of view: since warping a given frequency band results in a dilated or compressed band, without normalization the warped band may be perceived as louder or fainter, respectively. If the map θ is strictly increasing, then the warping operator is invertible and the action of the inverse is given by

$$\begin{aligned}\left[\mathfrak{U}_\theta^\dagger S\right](\omega) &= \left[\mathfrak{U}_\theta^{-1} S\right](\omega) = [\mathfrak{U}_{\theta^{-1}} S](\omega) \\ &= \sqrt{\frac{d\theta^{-1}}{d\omega}} S(\theta^{-1}(\omega)),\end{aligned} \tag{5}$$

where the symbol † denotes the adjoint operator, which is identical to the inverse in view of unitarity.

A spectral peak of the signal at $\omega = \omega_0$ may result into one or more peaks of the warped signal located at the roots Ω of the equation $\theta(\Omega) = \omega_0$, if any. If the map θ is one-to-one and onto then the peak at $\omega = \omega_0$ transforms into a peak at $\theta^{-1}(\omega_0)$. In this sense, frequency warping is a frequency dependent modulation technique, where each component sinusoid is modulated to a different frequency.

2.1. Warping Quasi-Sinusoidal Signals

It is instructive to investigate on how an amplitude modulated sinusoidal signal of the form

$$s(t) = g(t)e^{j\omega_0 t}, \tag{6}$$

where $g(t)$ is a real smooth envelope, is transformed by frequency warping. In this case we have

$$S(\omega) = G(\omega - \omega_0), \tag{7}$$

hence the Fourier transform of the warped signal is

$$\tilde{S}(\omega) = \sqrt{\frac{d\theta}{d\omega}} G(\theta(\omega) - \omega_0). \tag{8}$$

If the amplitude envelope $g(t)$ is narrowband then $G(\theta(\omega) - \omega_0)$ is nonzero only in a small neighborhood of $\theta^{-1}(\omega_0)$. Therefore, if the map is continuous and differentiable in this neighborhood, we can expand $\theta(\omega)$ in a Taylor series about $\omega = \omega_0$. Truncation to first order yields:

$$\begin{aligned}\theta(\omega) &\approx \theta(\theta^{-1}(\omega_0)) + \beta(\omega - \theta^{-1}(\omega_0)) \\ &= \omega_0 + \beta(\omega - \theta^{-1}(\omega_0)),\end{aligned} \tag{9}$$

where

$$\beta = \frac{d\theta}{d\omega}\Big|_{\omega=\theta^{-1}(\omega_0)} = \left[\frac{d\theta^{-1}}{d\omega}\Big|_{\omega=\omega_0}\right]^{-1}. \tag{10}$$

Substituting (9) into (8) and approximating, within the warped band, the first derivative of θ with the constant β obtains

$$\tilde{S}(\omega) \approx \sqrt{\beta} G(\beta(\omega - \theta^{-1}(\omega_0))). \tag{11}$$

Equation (11) shows that warping a narrowband signal is approximately equivalent to scaling the signal envelope by β and to remodulating to the warped frequency $\theta^{-1}(\omega_0)$. Indeed, from (11) and the Fourier scale theorem, we obtain:

$$\tilde{s}(t) \approx \frac{1}{\sqrt{\beta}} g\left(\frac{t}{\beta}\right) e^{j\theta^{-1}(\omega_0)t}. \tag{12}$$

Consequently, the frequency warped version of a wideband signal represented in terms of narrowband Gabor grains can be obtained approximately by individually scaling and remodulating the grains.

Another essential ingredient for the approximation of warping is dispersion. It is easy to see that the warped version of the shifted signal $s(t - \tau)$ has Fourier transform

$$[\mathfrak{F}\mathfrak{U}_\theta s(t-\tau)](\omega) = \sqrt{\left|\frac{d\theta}{d\omega}\right|} e^{-j\theta(\omega)\tau} S(\theta(\omega)). \tag{13}$$

Accordingly, each frequency component of the signal is time shifted by a different amount controlled by the warping map θ, acting as a multiplier of the time shift τ. If $s(t)$ is the narrowband signal in (6) then, within the same approximation as in (11), we have

$$[\mathfrak{F}\mathfrak{U}_\theta s(t-\tau)](\omega) \approx e^{-j\left[\beta\omega + \omega_0 - \beta\theta^{-1}(\omega_0)\right]\tau} \tilde{S}(\omega). \tag{14}$$

As the result of warping, the delayed enveloped sinusoid is shifted by $\beta\tau$. Hence, the warped group delay β acts as a multiplier of time shift. The phase correction term $\left[\beta\theta^{-1}(\omega_0) - \omega_0\right]\tau$ adjusts the phase delay of the unwarped sinusoid to that of the warped one. Indeed, it is easy to show that the approximately warped version of a running sinusoid amplitude modulated by a delayed envelope, i.e., of a signal of the form:

$$v(t) = g(t-\tau)e^{j\omega_0 t}, \tag{15}$$

is the following:

$$\tilde{v}(t) = \mathfrak{U}_\theta v(t) \approx \frac{1}{\sqrt{\beta}} g\left(\frac{t}{\beta} - \tau\right) e^{j\theta^{-1}(\omega_0)t}. \tag{16}$$

The last approximation can be written in the compact form:

$$\mathfrak{U}_\theta \mathfrak{M}_{\omega_0} \mathfrak{T}_\tau g(t) \approx \mathfrak{M}_{\theta^{-1}(\omega_0)} \mathfrak{T}_{\beta\tau} \mathfrak{D}_\beta g(t), \tag{17}$$

where $\mathfrak{M}_{\omega_0}$ is the modulation operator

$$\mathfrak{M}_{\omega_0} g(t) = e^{j\omega_0 t} g(t), \tag{18}$$

$\mathfrak{T}_\tau$ is the time-shift operator

$$\mathfrak{T}_\tau g(t) = g(t-\tau), \tag{19}$$

and $\mathfrak{D}_\beta$ is the dilation operator

$$\mathfrak{D}_\beta g(t) = \tfrac{1}{\sqrt{\beta}} g\left(\tfrac{t}{\beta}\right). \tag{20}$$

In other words, since (17) can be rewritten as follows:

$$\mathfrak{D}_\beta^{-1}\mathfrak{T}_{\beta\tau}^{-1}\mathfrak{M}_{\theta^{-1}(\omega_0)}^{-1}\mathfrak{U}_\theta\mathfrak{M}_{\omega_0}\mathfrak{T}_\tau g(t) \approx g(t), \tag{21}$$

the low-pass window $g(t)$ must be selected as an approximate eigenfunction of the unitary operator on the left hand side, with eigenvalue 1.

It is interesting to note that both the modulation and the dilation operator are particular cases of unitary warping operators, where the map θ is chosen as $\omega - \omega_0$ and $\beta\omega$, respectively. Indeed, using this fact and the commutation rule $\mathfrak{T}_{\beta\tau}\mathfrak{D}_\beta = \mathfrak{D}_\beta\mathfrak{T}_\tau$, one can show that (21) is equivalent to

$$\mathfrak{U}_\psi\mathfrak{T}_\tau g(t) \approx \mathfrak{T}_\tau g(t) \tag{22}$$

with

$$\psi(\omega) = \theta\left(\tfrac{\omega}{\beta} + \theta^{-1}(\omega_0)\right) - \omega_0. \tag{23}$$

In other words, the time-shifted window must be close to an eigenfunction of the incremental warping operator $\mathfrak{U}_\psi$.

3. APPROXIMATE FREQUENCY WARPING THROUGH GABOR FRAMES

By means of Gabor expansions [7, 6], signals are represented in terms of a collection of windowed sinusoids:

$$s(t) = \sum_{q,n\in\mathbf{Z}} S_{q,n} g_{q,n}(t), \tag{24}$$

where

$$g_{q,n}(t) = \mathfrak{M}_{2\pi qa}\mathfrak{T}_{n\tau} g(t) = e^{j2\pi qat} g(t-n\tau);\ q,n\in\mathbf{Z}, \tag{25}$$

in which q is the frequency index and n the time index. The representative elements are obtained by time-shifting and modulating a unique window function $g(t)$. The representation is complete provided that the frame operator

$$\mathfrak{P} s(t) = \sum_{q,n\in\mathbf{Z}} \langle s, g_{q,n}\rangle\, g_{q,n}(t) \tag{26}$$

is invertible, where the symbol $\langle,\rangle$ denotes scalar product in $\mathbf{L}^2(\mathbf{R})$. This is true if there exist two finite non-zero constants A and B such that

$$A \leqslant \sum_{q,n\in\mathbf{Z}} |\langle s, g_{q,n}\rangle|^2 \leqslant B, \tag{27}$$

in which case (25) is said to be a frame and the expansion coefficients $S_{q,n}$ in (24) can be computed – not uniquely in the general case – as the scalar products

$$S_{q,n} = \langle s, \hat{g}_{q,n}\rangle, \tag{28}$$

where $\hat{g}_{q,n} = \mathfrak{P}^{-1} g_{q,n}(t)$ is the dual frame. For the Gabor frame (27) can only be satisfied if the time-frequency sampling grid is sufficiently fine, i.e., if $a\tau \leqslant 1$.

The definition of frame (27) and of frame operator (26) is independent on the way the frame elements $g_{q,n}(t)$ are generated, i.e., it extends to non-Gabor frames in which the frame elements are not generated by time-shift and modulation. For the Gabor frame one can show that

$$\hat{g}_{q,n}(t) = \mathfrak{M}_{2\pi qa}\mathfrak{T}_{n\tau}\hat{g}(t) = e^{j2\pi qat}\hat{g}(t-n\tau);\ q,n\in\mathbf{Z}, \tag{29}$$

i.e., the dual Gabor frame is also obtained by time-shifting and modulating a unique dual window function $\hat{g}(t)$, so that (28) takes on the form of a short-time Fourier transform with inverse (24).

Any unitary operation on the frame results in a new frame with the same frame bounds A and B [13]. In particular, when applied to a Gabor frame, the unitary warping operator $\mathfrak{U}_\theta$ generates the frequency warped frame and dual frame

$$\begin{aligned} \tilde{g}_{q,n}(t) &= \mathfrak{U}_\theta\mathfrak{M}_{2\pi qa}\mathfrak{T}_{n\tau} g(t), \\ \tilde{\hat{g}}_{q,n}(t) &= \mathfrak{U}_\theta\mathfrak{M}_{2\pi qa}\mathfrak{T}_{n\tau}\hat{g}(t). \end{aligned} \tag{30}$$

A discrete-time version of warped frame was introduced in [14] for the non-uniform time-frequency representation of signals. As we showed in the previous Section, the warped Gabor frame and its dual are not Gabor frames. Unless the warping map is linear, there is no exact commutation rule between warping and time-shift operators. However, reasoning as in (17), one can show that

$$\tilde{g}_{q,n}(t) \approx \mathfrak{M}_{\theta^{-1}(2\pi qa)}\mathfrak{T}_{n\beta_q\tau}\mathfrak{D}_{\beta_q} g(t), \tag{31}$$

where

$$\beta_q = \left.\frac{d\theta}{d\omega}\right|_{\omega=\theta^{-1}(2\pi qa)} = \left[\left.\frac{d\theta^{-1}}{d\omega}\right|_{\omega=2\pi qa}\right]^{-1}, \tag{32}$$

provided that $g(t)$ is sufficiently smooth and narrowband, and similarly for the dual frame.

In this paper we will not focus on the unitarily equivalent warped frame representation. Rather, we seek approximations of the warping operator through Gabor frame representation. The main idea is that when (31) holds and the signal is represented as in (24) then

$$\begin{aligned} \mathfrak{U}_\theta s(t) &= \sum_{q,n\in\mathbf{Z}} S_{q,n}\mathfrak{U}_\theta g_{q,n}(t) \\ &\approx \sum_{q,n\in\mathbf{Z}} S_{q,n}\mathfrak{M}_{\theta^{-1}(2\pi qa)}\mathfrak{T}_{\beta_q\tau}\mathfrak{D}_{\beta_q} g(t). \end{aligned} \tag{33}$$

Notice that (33) is a peculiar "irregular" Gabor-like expansion in which the windows are differently scaled by the dilation operators $\mathfrak{D}_{\beta_q}$ and the modulation frequencies $\theta^{-1}(2\pi qa)$ are not harmonically related. Moreover, the expansion coefficients $S_{q,n}$ are obtained as in (28) by computing the scalar product of the signal with the unwarped dual Gabor frame $\hat{g}_{q,n}(t)$.

An alternate (dual) scheme consists, in principle, in warping the signal before computing the Gabor coefficients and using these coefficients for the expansion on a conventional Gabor frame. Thus, one can compute the coefficients

$$\tilde{S}_{q,n} = \langle \mathfrak{U}_\theta s, \hat{g}_{q,n}\rangle = \left\langle s, \mathfrak{U}_\theta^\dagger \hat{g}_{q,n}\right\rangle. \tag{34}$$

As shown in the rightmost term of (34), this process is unitarily equivalent to computing the scalar product of the signal over the inversely warped dual frame:

$$\tilde{\hat{g}}_{q,n}(t) = \mathfrak{U}_{\theta}^{\dagger}\hat{g}_{q,n}(t) = \mathfrak{U}_{\theta^{-1}}\mathfrak{M}_{2\pi qa}\mathfrak{T}_{n\tau}\hat{g}(t). \qquad (35)$$

Summarizing the results of this Section, approximate frequency warped can be computed by means of Gabor-like expansions in which either the frame elements are approximately warped by scaling and frequency dependent modulation or the dual frame elements are approximately inversely warped by scaling and frequency dependent modulation.

The resulting piecewise warping map approximation is shown in Figure 2 for a very reduced number of bands. There, a warping curve is approximated by tangent linear segments in partly overlapping bands. The edges and centerbands of the ideal uniform bandwidth frame elements can be read on the ordinates axis. These are transformed according to the inverse map θ^{-1} into non-uniformly spaced frequencies as read on the abscissae axis (dotted lines). As a result of the approximation, the band edges are transformed according to the tangent lines and can be traced in the figure as dashed lines.

We remark that, according to the local convexity of the warping map, the bands resulting from the local scaling approximation of warping may either overlap in the frequency domain or there can be gaps. However, as the number of bands increases, the overlapping portions or the gaps become less and less pronounced. Other piecewise linear approximations are possible, e.g., by tracing chord segments joining the band edges. The one we selected exactly warps the centerband frequencies.

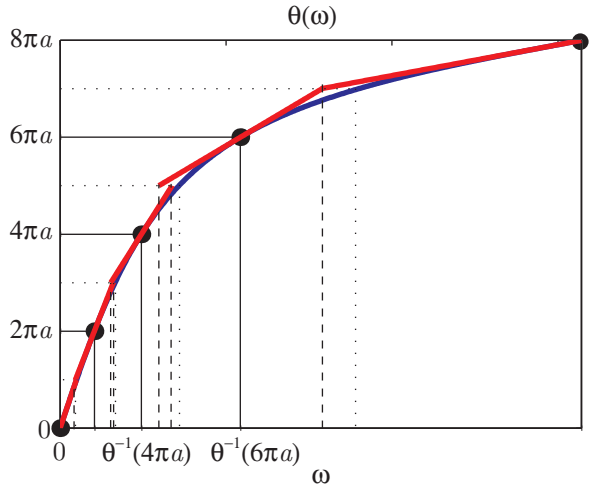


Figure 2: *Piecewise linear approximation of warping map with a very small set of points: thin solid lines denote center bands and dotted lines denote band edges(initial and warped).*

4. DISCRETE-TIME FREQUENCY WARPING

The discrete-time counterpart of the frequency warping approximation easily follows from the continuous time version discussed in the previous sections. The discrete-time Gabor set requires only a finite number of bands M to cover the normalized frequency range $\omega \in [-\pi, +\pi)$, with frequency resolution $a = 2\pi/M$. The discrete-time Gabor frame elements $g_{q,n}(r)$ are obtained by shifting and modulating a unique window sequence $g(n)$, where only integer shifts multiple of an integer $N \leq M$ are allowed, thus

$$g_{q,n}(r) = \mathfrak{M}_{\frac{2\pi q}{M}}\mathfrak{T}_{nN}g(r) = e^{j\frac{2\pi q}{M}r}g(r - nN), \qquad (36)$$

for $q = 0, 1, ..., M-1$ and $n \in \mathbf{Z}$, and similarly for the dual frame $g_{q,n}(r)$.

A few remarks on the warping maps are however necessary. In fact, since frequency warping transforms bandwidths, the restriction of an invertible analog frequency map to the interval $[-\pi, +\pi)$ is not necessarily one-to-one and onto over this interval. For a strictly increasing map mapping zero frequency to zero frequency one needs to require that $\pm\pi$ is mapped to $\pm\pi$. If the map maps $[-\pi, +\pi]$ one to one onto a smaller interval then aliasing-like phenomena occur, similar to downsampling, unless the original signal sequence is suitably bandlimited to the smaller interval $\theta([-\pi, +\pi))$. Vice versa, if the map maps $[-\pi, +\pi]$ one to one onto a larger interval then spectral replication occurs, similar to upsampling, unless the domain of the map is restricted to the smaller interval $\theta^{-1}([-\pi, +\pi))$. In other words, in order to avoid both aliasing and imaging one needs to restrict both domain and co-domain of the warping map to the interval $[-\pi, +\pi]$, keeping in mind that there can be frequencies that are not mapped by any frequency or frequencies that do not map to any frequency. In the latter case, only the scalar products of the signal with a subset of the Gabor frame elements, with respect to the frequency index q, are required in order to compute warping.

The choice of the analysis and synthesis windows, respectively $\hat{g}(n)$ and $g(n)$, is arbitrary, provided that the discrete-time counterpart of (27) is satisfied. Ideally, the synthesis window should match eigenfunctions of the incremental warping operator $\mathfrak{U}_{\psi}$ as in (22). However, the Fourier transform of an eigenfunction of the unitary warping operator corresponds to the square root of the derivative of an eigenfunction of the composition operator $\mathfrak{C}_{\psi}$, up to an additive constant. Using the theory developed in [15], we were able to prove that the only eigenfunctions of $\mathfrak{U}_{\psi}$ with eigenvalue 1 are constant, corresponding to a Dirac pulse at $\omega = 0$ in the frequency domain. In order to approach this shape, we can select the window to be narrowband lowpass, which leads to conventional window design in short-time spectral analysis. We let M be an integer multiple of N, i.e., $M = KN$, with K a small integer (usually $K = 2$ or 3), and determine a length M lowpass symmetric window $g(n)$ such that

$$\sum_{n\in\mathbf{Z}} g^2(r - nN) = \frac{1}{KM}. \qquad (37)$$

In this case, by choosing identical analysis and synthesis windows: $g(n) = \hat{g}(n)$, one obtains a tight frame, i.e., a frame with bounds $A = B = 1$ in (27). This is similar to orthogonal bases, however, the frame is complete but not a basis. A popular choice is

$$g^2(r) = \frac{1 - \cos\left(\frac{2\pi r}{M}\right)}{KM};\ r = 0, 1, ..., M-1, \qquad (38)$$

i.e., $g^2(r)$ is the Von Hann window and

$$g(r) = \sqrt{\frac{2}{KM}}\sin\left(\frac{\pi r}{M}\right);\ r = 0, 1, ..., M-1 \qquad (39)$$

is the sine window, which we will enforce in this paper. One of the advantages of this choice is that the sinusoidal form of the window

leads to a single frequency term, which warps to another single frequency term. Therefore, except for terms deriving from the finite length of the window, the shape of the warped window is very close to that of a dilated window.

Computation of the expansion coefficients

$$S_{q,n} = \langle s, g_{q,n} \rangle = \sum_{r \in \mathbf{Z}} s(r) e^{-j\frac{2\pi}{M}qr} g(r - nN) \qquad (40)$$

can be performed using the analysis section of the filter bank structure in Figure 3, which corresponds to the analysis section of a phase vocoder [16, 17, 8], with impulse responses

$$g_q(r) = e^{j\frac{2\pi}{M}qr} g(M - r) = e^{j\frac{2\pi}{M}qr} g(r). \qquad (41)$$

A delay of $K-1$ samples is introduced in the coefficient sequence $S_{q,r}$ in order to make computation causal.

The approximate synthesis of the warped signal requires a discrete counterpart of (33). The discrete-time window can be considered as the samples of a continuous-time function $g(t)$ taken at unit sampling rate. In order to form the discrete-time warping synthesis windows one can apply the dilation operation $\mathfrak{D}_{\beta_q} g(t)$ to the continuous-time window and then sample at unit sampling rate. This operation yields the window

$$h_q(r) = \sqrt{\frac{1}{\beta_q}} g\left(\frac{r}{\beta_q}\right). \qquad (42)$$

However, the result is sharper if only the window length M is modulated to some other integer M_q. To the purpose we let M_q be the closest integer to $\beta_q M$ and we let

$$h_q(r) = \sqrt{\frac{M}{M_q}} g\left(\frac{rM}{M_q}\right);\ r = 0, 1, ..., M-1. \qquad (43)$$

Similarly, for the discrete-time counterpart of the shift operator $\mathfrak{T}_{\beta_q \tau}$, we enforce the integer shifts $nN_q = nM_q/K$. Both dilation and shifting operations are quantized in the discrete-time warping algorithm, which is a source of error that can be controlled by choosing a sufficiently large window length M and a sufficiently small overlap factor K. This is preferable with respect to the non-integer choice, which causes amplitude modulation since the window would not satisfy the overlap add identity (37). In this form, the approximate warped signal can be obtained as output of the synthesis filter bank in Figure 3. There, the analysis coefficients are phase corrected, upsampled with a different upsampling rate in each channel and interpolated by unequal length modulated windows

$$\tilde{g}_q(r) = \mathfrak{M}_{\theta^{-1}\left(\frac{2\pi q}{M}\right)} h_q(r) = e^{-j\theta^{-1}\left(\frac{2\pi q}{M}\right)r} h_q(r). \qquad (44)$$

The phase correction terms

$$\gamma_q = N_q \theta^{-1}\left(\tfrac{2\pi q}{M}\right) - \tfrac{2\pi q N}{M}, \qquad (45)$$

which are the discrete analogues of (14), originate from the filter bank implementation of the phase vocoder, where the modulating terms in the frame elements need to be time shifted for both the analysis and synthesis procedures to be put in the form of (resampled) convolution. To the purpose, we use the following commutation rule:

$$\mathfrak{M}_{\omega_0} \mathfrak{T}_\tau = e^{j\omega_0 \tau} \mathfrak{T}_\tau \mathfrak{M}_{\omega_0} \qquad (46)$$

in both the analysis and synthesis. While in conventional phase vocoders the phase correction factors cancel out, in reason of the heterogeneous analysis and synthesis sections these factors are indeed necessary for correct computation in the approximate warping filter bank scheme.

An alternate structure, with similar properties, to that shown in Figure 3 can be worked out from (34) and (35). It consists of an approximate inverse warping analysis section, followed by a conventional phase vocoder synthesis section. Furthermore, an extension to rational numbers N_q and M_q is possible, which achieves a smaller error but requires a more costly implementation in terms of upsamplers and downsamplers by possibly high factors.

5. PERFORMANCE

In this Section we give a brief account of the performance of the approximate warping algorithm, in terms of operation count and numerical and perceptual error.

5.1. Computational Cost

The analysis section of the approximate warping filter bank in Figure 3 can be efficiently implemented with a length M FFT, with a cost of $O(M \log M)$ operations, which are repeated every N input samples, for a total rate of $O(K \log M)$ operations per sample. The synthesis filter bank section cannot be efficiently computed by means of FFT since the modulating frequencies of the warped frame elements are not harmonically related. For real signals, only half of the filters in the bank must be computed since the other half are complex conjugated versions applied to complex conjugated coefficient sequences. Therefore, a set of $M/2$ real impulse responses can be derived by combining each pair of complex conjugates sections of index q and $M-q$. For each expansion coefficient, the generic section q filter generates N_q output samples; the other $M_q - N_q = (K-1)N_q$ samples need to be prepared for the overlap with subsequent outputs in the overlap-add scheme. This operation requires M_q multiplies of the coefficient $S_{q,r}$ times the corresponding precomputed synthesis window. Thus, each filter section requires $M_q/N_q = K$ operations per output sample. For M even there are $M/2+1$ sections and for M odd there are $\lfloor M/2 \rfloor + 1$ sections. In each length N_q output segment there are K slices of shifted windows to be added together. Therefore, the total cost of the complete synthesis section is $(\lfloor M/2 \rfloor + 1)\, K$ multiplies and $\lfloor M/2 \rfloor\, (K-1)$ adds per sample. Notice that in this computational scheme the phase correction factors are not needed as they are embedded in the modulated windows both in the analysis and in the synthesis.

For a finite-length input signal, the overall cost of the approximate frequency warping structure grows linearly. This should be compared with the non-causal allpass cascade structures in classical computation of the Laguerre transform for frequency warping [18, 14], whose complexity grows quadratically with the number of samples.

5.2. Numerical and Perceptual Error

A direct estimate of the warping approximation error bounds provides the following characteristics: both the ℓ^2 error norm and the absolute error decrease as $M^{-3/2}$ as M grows. Therefore, the error can be reduced by choosing a sufficiently wide window. Moreover, due to the fact that the approximation of the delays tends to

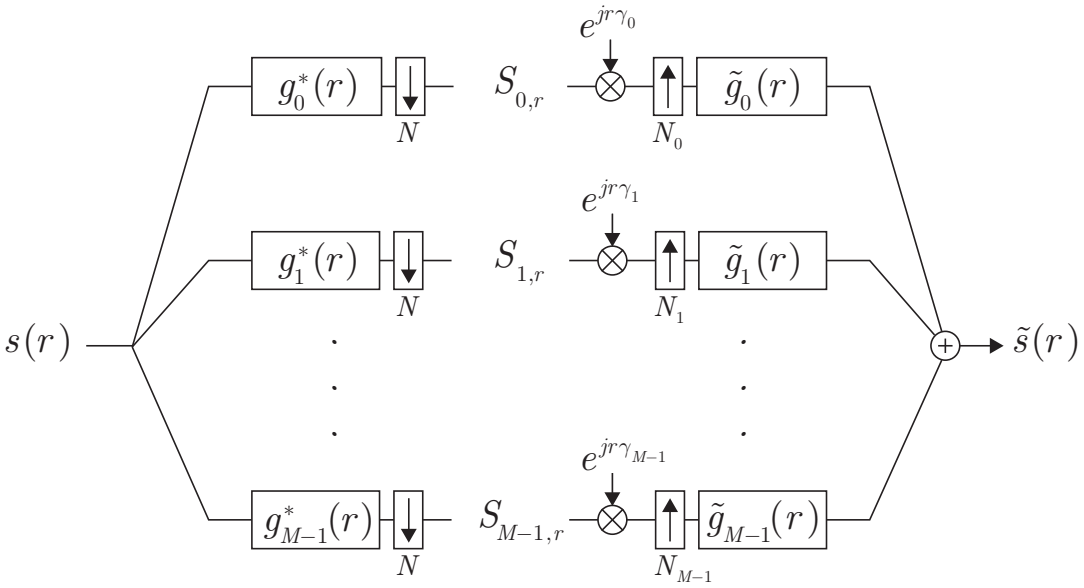


Figure 3: *Efficient warping structure based on Gabor filter bank for the analysis and approximated warped Gabor filter bank for the synthesis.*

cumulate in time, the error grows with the length of the input signal. With this respect, for fixed window length M, our estimates provide an error trend of $O(L^{5/4}/M^2)$ as the length L of the signal grows. For example, for a window length of 2400 samples, the error remains confined within 10% for about 1 s at 44.1 kHz sampling rate. The error is also proportional to the sum of the coefficients β_q, showing that the error depends on the steepness of the warping curve. Moreover, the influence of the overlap factor over the error decays modestly as $K^{-1/2}$. However, since N decreases as K grows, the heavier quantization of the delays and of the window shifts tends to increase the error. Therefore, from the error point of view increasing K is not beneficial.

We extensively tested the theoretical error estimates against the numerical error. In this task we employed a number of sources like noise and tones of musical instruments with sharp or slow attack and decay. In the comparison we used the one-parameter family of Laguerre maps, as only in this case an exact algorithm to compute warping is available. The experimental results confirm the theoretical estimates and show proportionality with the given trends by small constants, confirming that the theoretical estimates provide worst case bounds.

Additionally, we evaluated the perceptual error introduced by warping. For the purpose, for a set of instrument sounds, we evaluated the error of the approximation with respect to Laguerre maps. Using a procedure similar to that found in MPEG coders, we computed the masking curves of the signal over the error. As the error is coherent with the signal, masking is likely to occur. This is due to the fact that the most relevant part of the error originates from the approximation of frequency dependent delays as constant within narrow frequency bands and from their quantization. In our results, the approximation error rarely exceeds the masking threshold. In sounds of approximately 1 s duration and using a window length of $M = 2400$ samples, the masking threshold overshooting occurred in less than 1% of the frames, mostly located at the attack and at the decay segments of the tones.

6. APPLICATIONS

The frequency warping effect introduces a coloration in the signal in which harmonically related partial can become more or less inharmonic, with consequent beating and floating. The effect is per se interesting and can be applied to the sounds of several instruments and, especially, strings. Examples employing the proposed algorithm on a variety of sounds, together with a comparison with the exact algorithm in the available cases (Laguerre maps) can be found at the following URL: `http://www.itn.liu.se/~giaev/soundexamples.html`.

With properly selected maps, frequency warping can be used in physical models as an alternative to dispersive delay lines. In fact, a dispersive delay line can be considered as a warped delay line in which the elementary delay (or a group of them) is replaced by an allpass filter. The structure is therefore equivalent to a delay line cascaded with a warping section, provided that all the inputs are inversely warped. A dispersive waveguide may contain several dispersive digital lines, each going from the excitation input to the boundaries or vice versa and to and from the output pick-up point. Often the inputs are control signals that can be either prewarped

off-line or, in the case of noise, are unaffected by warping. Each dispersive chunk is formed by a chain of allpass filters, or, what is equivalent, by a high order all-pass filter. The design of heterogeneous length allpass filters simulating the proper dispersion is very hard and still an open problem [12]. Rather, it may be convenient to simulate a non-dispersive waveguide, moving dispersion outside the closed loop, which is accomplished by warping the output signal.

In contrast to the one parameter family of Laguerre maps in allpass cascade implementations, the proposed warping algorithm leaves ample freedom in the choice of warping map. In fact, fixed the window length M, both the discrete set of warped frequencies $\omega_q = \theta^{-1}(2\pi q/M)$ and the slopes β_q can be arbitrarily selected. The latter control the scale of the warped signal about the corresponding frequency ω_q, where large values of β_q generate longer signals. This property can be used in digital audio warping effects in order to change the decay rates of the output signal in a frequency dependent fashion.

While the cascade allpass structure for Laguerre warping are strongly non-causal and impractical for real-time computation, the proposed warping algorithm lends itself to real-time computation. However, the limitation of our algorithm toward real-time is that the upsampling rates N_q in each channel of the synthesis structure should never be smaller than the corresponding downsampling rate N of the analysis section, otherwise data would be missing for the computation of the current sample. Since the ratio $N_q/N \approx \beta_q$, then (see (32)) a condition for real-time operation is that the first derivative of the warping map should not be smaller than 1. Warping maps having this characteristic bring frequencies in $[-\pi, +\pi]$ to frequencies in a smaller interval around zero frequency.

In the synthesis by physical models, the previous limitation can be circumvented by calibrating the non-dispersive waveguide to a much higher pitch than the target one, so that the tone is brought back to the desired height by warping. If the warping map has a specific form $\hat{\theta}(\omega)$, as dictated by physical laws or by experimental measures, one can easily transform this into a map having derivative greater than one simply by multiplying it by a constant $\alpha > 1$. With the new map $\theta(\omega) = \alpha\hat{\theta}(\omega)$, a frequency ω_0 warps to frequency $\tilde{\omega}_0 = \theta^{-1}(\omega_0/\alpha)$. Therefore, in order to achieve the desired frequency $\omega_d = \hat{\theta}^{-1}(\omega_0)$ one simply needs to start with frequency $\alpha\omega_0$. The linearity of this relationship ensures that the relationships among the desired frequencies, e.g., the harmonics to be warped to inharmonic partials with a given law, is not altered. The only limitation is therefore the reduced frequency range due to the $[-\pi, +\pi]$ clipping of the scaled warping map. This can be handled by oversampling.

In digital audio effects, the above limitation toward real-time may be too severe. However, a workaround is possible by embedding pitch-shifting in the modified phase vocoder scheme. This is indeed directly possible by frequency bin reassignment and by phase correction in the analysis section [19, 20].

The latency of the approximate warping scheme is proportional, via the sampling interval, to the window shift integer parameter N. The latter is the minimum number of samples in order to produce a set of coefficients, assuming that the previous $(K-1)N$ samples are known or zero. For a fixed window length M, one can reduce N by increasing the overlap factor K. However, as we have seen in Section 5, reducing N increases the computational cost and the quantization error of the delays and window shifts in the synthesis section. This results in lower adherence to the established warping rule but in no other annoying effect. Therefore, a trade-off between latency and precision needs to be achieved. For $K = 2$ and $M = 2200$, latency is estimated at about $25\ ms$ at a sampling rate of 44.1 kHz.

7. CONCLUSION

In this paper we introduced an efficient scheme for frequency warping audio signals, which overcomes several limitations of allpass chain based systems, notably those of the restricted family of achievable maps and the non-causality of the computational structure. The approximate warping algorithm was shown to introduce a negligible error, both numerical and perceptual, which can be controlled by proper choice of the length of the window. With some limitations and workarounds, the algorithm has a real-time implementation and it can be suitable as a digital audio effect or in the synthesis by physical models.

8. REFERENCES

[1] A. Constantinides, "Spectral transformations for digital filters," *Proc. Inst. Elec. Eng.*, vol. 117, pp. 1585–1590, Aug. 1970.

[2] GRM-INA, "FreqWarp plugin," `http://www.grmtools.org/quicktour/vstqtst/warp.html`.

[3] Prosoniq, "π-Warp plugin," `http://products.prosoniq.com/cgi-bin/register?service=showdetail\&refno=36`.

[4] G. Evangelista, "The short-time Laguerre transform: a new method for real-time frequency warping of sounds," in *Proc. of the Int. Computer Music Conf. (ICMC 2000)*, Berlin, Germany, Aug. 2000, pp. 380–383.

[5] G. Evangelista, "Time and frequency warping of musical signals," in *Digital Audio Effects*. Wiley, Chichester, UK, 2002.

[6] D. Gabor, "Theory of communication," *J. IEE*, vol. 93, no. 26, pp. 429–457, 1946.

[7] J. J. Benedetto and D. F. Walnut, "Gabor frames for L^2 and related spaces," in *Wavelets: Mathematics and Applications*, J.J. Benedetto and M.W. Frazier, Eds., pp. 97–162. CRC Press, 1994.

[8] M. E. Portnoff, "Time-frequency representations of digital signals and systems based on short time Fourier analysis," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 28, pp. 55–69, 1980.

[9] G. Evangelista and S. Cavaliere, "Discrete frequency warped wavelets: Theory and applications," *IEEE Trans. on Signal Processing*, vol. 46, no. 4, pp. 874–885, April 1998, Special issue on Theory and Applications of Filter Banks and Wavelets.

[10] G. Evangelista and S. Cavaliere, "Audio effects based on biorthogonal time-varying frequency warping," *EURASIP Journ. on Applied Signal Processing*, vol. 2001, no. 1, pp. 27–35, March 2001.

[11] I. Testa, G. Evangelista, and S. Cavaliere, "Physically inspired models for the synthesis of stiff strings with dispersive

waveguides," *EURASIP Journ. on Applied Signal Processing*, vol. 2004, no. 7, pp. 964–977, July 2004, special issue on Model-Based Sound Synthesis.

[12] J. S. Abel and J. O. Smith, "Robust design of very high-order allpass dispersion filters," in *Proc. of the Int. Conf. on Digital Audio Effects (DAFx-06)*, Montreal, Quebec, Canada, Sept. 18–20, 2006, pp. 13–18.

[13] R. G. Baraniuk and D. L. Jones, "Unitary equivalence : A new twist on signal processing," *IEEE Trans. Signal Processing*, vol. 43, no. 10, pp. 2269–2282, Oct. 1995.

[14] A. V. Oppenheim, D. H. Johnson, and K. Steiglitz, "Computation of spectra with unequal resolution using the Fast Fourier Transform," *Proc. IEEE*, vol. 59, pp. 299–301, Feb. 1971.

[15] P. S. Bourdon and J. H. Shapiro, "Mean growth of Koenigs eigenfunctions," *J. American Math. Soc.*, vol. 10, no. 2, pp. 299+325, April 1997.

[16] M. Dolson, "The phase vocoder: A tutorial," *Comp. Music J.*, vol. 10, no. 4, pp. 14–27, 1986.

[17] J. L. Flanagan and R. M. Golden, "Phase vocoder," *Bell System Tech. J.*, pp. 1493–1509, November 1966.

[18] P. W. Broome, "Discrete orthonormal sequences," *J. Assoc. Comput. Machinery*, vol. 12, no. 2, pp. 151–168, 1965.

[19] J. Laroche and M. Dolson, "New phase vocoder technique for pitch-shifting, harmonizing and other exotic effects," in *Proc. IEEE Workshop on Appl. Signal Proc. Audio and Acoustics*, New Paltz, NY, 1999.

[20] M. Puckette, "Phase-locked vocoder," in *Proc. IEEE ASSP Conf. Appl. Signal Proc. Audio and Acoustics*, New Paltz, NY 1995.

SHORT-TIME WAVELET ANALYSIS OF ANALYTIC RESIDUALS FOR REAL-TIME SPECTRAL MODELLING

Jeremy J. Wells and Damian T. Murphy

Audio Lab, Intelligent Systems Group, Department of Electronics
University of York, YO10 5DD, UK
{jjw100|dtm3@}ohm.york.ac.uk

ABSTRACT

This paper describes an approach to using compactly supported spline wavelets to model the residual signal in a real-time (frame-by-frame) spectral modelling system. The outputs of the model are time-varying parameters (gain, centre frequency and bandwidth) for filters which can be used in a subtractive resynthesis system.

1. INTRODUCTION

Extraction of the sinusoidal part of an audio signal, leaving a residual, is a common approach to spectral modelling [1]. Whereas the sinusoidal part of the signal consists of long-term, narrow-band components, the residual is comprised of both long- and short-term broad-band components. Systems have been proposed that separate these two types of residual component, classifying them as transients or noise, such as in [2]. A multiresolution approach to residual modelling, such as that offered by wavelets, can enable the good time localisation required for transients, along with the generality required for more stationary components. This paper introduces such an approach.

The analysis system described here was developed for use in a real-time spectral modelling system. In this context real-time means frame-by-frame; decisions about model parameters are made, and time-domain resynthesis is executed, within the current frame to minimise the delay between input and output. Because it produces complex analysis data it is possible to obtain estimates for the centre frequency of components. Also, through use of the wavelet splitting method which is used in wavelet packet decomposition, the bandwidth of components can be estimated. This allows the resynthesis filters to adapt their time-frequency localisation properties to the analysed signal. B-spline wavelets are used since they also offer control over the time-frequency localisation of the analysis filters.

Whilst critically sampled orthogonal wavelets are often useful in situations where sparseness in the analysis data is required, for analysis-modelling-transformation applications over-complete (redundant) wavelet representations are usually more desirable. However such representations come at greater computational cost and highly redundant analysis may well be prohibitively expensive in a real-time application where the analysis-modelling-transformation-resynthesis cycle for a single frame must take less time than for that frame to be played out and the next frame to be acquired. To offer some mediation between cost and redundancy a 'partially decimated' transform is used where the amount of decimation can be controlled by the user (and potentially by the system in response to other processing demands).

Section 2 of this paper provides an introduction to the existing literature that describes B-spline wavelets and their properties. Section 3 gives an overview of the context in which they are used here and the spectral subtraction method used to obtain the residual. Section 4 describes the complex wavelet system employed while Section 5 assesses its cost and proposes partial decimation as a way of offering flexibility in this regard. Section 6 presents a method for estimating the bandwidth of components of the residual. Section 7 briefly describes a context in which the system is employed.

2. B-SPLINE WAVELETS

This section summarises the existing literature on B-splines and their associated wavelets. For further information the reader is directed to the references cited, particularly [3], [4] and [5].

A B-spline ('basis' spline) curve through a set of points consists of the linear combination of shifted B-spline basis functions of a given order. A zeroth order spline curve is constructed from a series of constant functions at the height of each data point. A first order spline curve is constructed from a series of straight lines that join each data point. A second order spline curve is constructed from a series of quadratic functions that span three data points and so on, with each 'piece' of the curve having its own weighting coefficient, meaning that a function can be described by:

$$f(x) = \sum_{k \in Z} c(k) \beta^m (x - k) \tag{1}$$

where β^m is the B-spline curve of order m and $c(k)$ are the weighting coefficients. The zeroth order B-spline is given by

$$\beta^0(x) = \begin{cases} 1, & -\frac{1}{2} < x < \frac{1}{2} \\ \frac{1}{2}, & |x| = \frac{1}{2} \\ 0, & \text{elsewhere} \end{cases} \tag{2}$$

Higher orders are obtained by repeated (m times) convolution of the zeroth order B-spline. For example the cubic (third order) B-spline is obtained by convolution of the zeroth order with itself three times. An order m B-spline can be found directly (without convolution) from:

$$\beta^m(x) = \frac{1}{m!} \sum_{k=0}^{m+1} \binom{m+1}{k} (-1)^k \left(x - k + \frac{m+1}{2} \right)_+^m \tag{3}$$

where

$$(x)_+^m = \begin{cases} x^m, x \geq 0 \\ 0, x < 0 \end{cases} \quad (4)$$

is the one-sided power function [3]. It should be noted that, for high order splines, there are stability problems when using (3) with finite precision. A version of (3) which exploits the symmetry inherent in the repeated convolution of a box function is proposed and used here. This is given by

$$\beta^m(x) = \frac{1}{m!}\sum_{k=0}^{m+1}\binom{m+1}{k}(-1)^k\left(-|x| - k + \frac{m+1}{2}\right)_+^m \quad (5).$$

The constant (zeroth order) B-spline basis leads to a model which is not continuous (since it is piecewise constant). The first order basis offers a continuous (piecewise linear) underlying model but it is not smooth since the first derivative is not continuous. The second order (quadratic) basis is continuous and smooth but its rate of change of curvature (second derivative) changes in a piecewise constant fashion. The third order (cubic) basis exhibits the 'minimum curvature property' since the second derivative is continuous and so for many applications the cubic B-spline is considered the most appropriate underlying continuous piecewise function. However, if better frequency localisation is required (at the cost of poorer time localisation) then the B-spline order can be increased.

For the *m* order B-spline wavelet transform the scaling and wavelet functions and their associated discrete filter sequences are given by

$$\beta^m(x/2) = \sum_{n=-\infty}^{\infty} u_2^m[n]\beta^m(x-n) \quad (6)$$

$$\psi(x/2) = \sum_{n=-\infty}^{\infty} g[n]\beta^m(x-n) \quad (7)$$

where $u_2^m[n]$, the interpolation (approximation) filter, is the binomial kernel of order *m* given by

$$u_2^m[n] = \begin{cases} \frac{1}{2^m}\binom{m+1}{n}, & 0 \leq m \leq n+1 \\ 0, & \text{otherwise} \end{cases} \quad (8)$$

and *g*[*n*], the wavelet (detail) filter, is given by

$$g[n] = u_2^m[n] * \left((-1)^m u_2^m[n]\right) * \left((-1)^m b^{2m+1}[n]\right) \quad (9)$$

where b^m is the *m*th order B-spline sampled at the integers [4], [6]. Figure 1 shows the wavelet functions associated with B-splines of order zero (the Haar wavelet), one, three and twenty, at scale one.

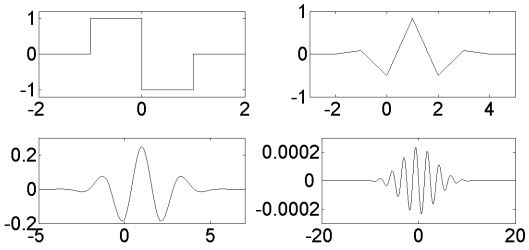


Figure 1: *Underlying wavelet functions for B-splines of different orders. The horizontal axis values are samples.*

As the order of the B-spline increases so the shape of the wavelet function tends to a modulated Gaussian (Gabor function) which has optimum time-frequency localisation properties. For a cubic B-spline it has been shown that the error in approximating a Gabor function is less than 3% and the localisation is within 2% of the optimum [5]. Unfortunately only the zeroth B-spline wavelet transform is energy preserving. At higher orders the wavelet and scaling filters are not the power complements of each other. Figure 2 shows the magnitude responses of these filters for orders zero, one and three. This lack of orthogonality can be overcome by over-sampling in both time (partial or no decimation) and scale domain (parallel transforms of the input at different sample rates) and by taking account of the approximate Gaussian shape of the filters in the Fourier domain. Knowledge of the filter shape, along with estimation of a component's width and centre frequency, allows for magnitude correction of estimates [8].

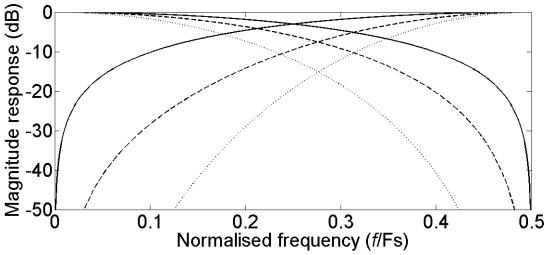


Figure 2: *Magnitude response of wavelet and scaling filters at orders zero (solid line), one (dashed) and three (dotted).*

3. DERIVATION OF THE RESIDUAL

The real-time spectral modelling system in which the analysis method described here is used is discussed in [7] and described in detail in [8]. The system models and synthesizes the sinusoidal part of monophonic audio signals as successive non-overlapping frames with piecewise quadratic phase and piecewise linear amplitude on a frame by frame basis. Only the phase is aligned between frames for continuing sinusoids; discontinuities in frequency and amplitude at synthesis frame boundaries are usually very small due to the high-accuracy analysis method employed.

The Spectral Modelling Synthesis system (SMS, see [1]) uses time domain subtraction to produce the residual; the entire sinusoidal

signal is synthesized and subtracted from the original input signal. An advantage of this approach is that having a time domain representation of the residual means that spectral analysis of it can be performed with optimised parameters, such as a shorter analysis frame, for what is assumed to be a stochastic signal. In a real-time system which produces output from input on a frame by frame basis it is not possible to employ this approach. Unless there is no overlap between frames (only possible with a rectangular window) the synthesis and analysis frames will be of a different length and so short-time time domain subtraction is not available either. For this reason spectral subtraction is employed here to calculate the residual signal. Once this has been performed the data is finally transformed back to the time domain in analytic form after Hilbert transformation in the Fourier domain, ready for the complex B-spline wavelet analysis described in this paper.

An assumption of SMS is "that the residual is fully described by its amplitude and its general frequency characteristics. It is unnecessary to keep either the instantaneous phase or the exact spectral shape information" [9]. Augmentations of the SMS model to include a third signal component type (transients) acknowledge that this assumption is not valid in some cases [2]. Whilst it is the case that for long term stationary noise the phase spectrum does not contain important information the situation for short duration broad band (i.e. impulsive) components is that both the phase and magnitude are needed to retain perceptually relevant fast changing temporal detail. The spectral modelling technique used here for the residual is intended to be capable of capturing the temporal detail of transient components and the spectral resolution of longer term stochastic components. Since both the phase and magnitude of non-sinusoidal components remain intact after spectral subtraction, the inherent timing information contained within these components is passed onto the complex wavelet analysis combining both transient and long term noise in the one model.

Time domain subtraction is a straightforward and, provided the instantaneous frequencies and amplitudes of the sinusoids are well predicted by the model, effective operation. Spectral subtraction is a more complex process since individual sinusoidal components are not represented by individual points in the Fourier domain. Finite length windowing smears components into multiple bins and non-stationarity exacerbates this: frequency change widens the main lobe and amplitude change narrows the main lobe but increases the level of side lobes, increasing the spread of energy to distant bins. A single sinusoid is represented by a single complex number in the Fourier domain only in a very specific situation: a rectangular analysis window is used, the analysed sinusoid has stationary amplitude and frequency and its frequency coincides exactly with the centre of an analysis bin (i.e. the length of the analysis window is an integer multiple of the sinusoidal period).

In [10] a spectral subtraction technique was described which was developed for use in a transform based thresholding process, *Wavethresh*. This technique used knowledge of the window power spectrum to predict the contribution made to adjacent bins made by a stationary sinusoid for a given deviation of the sinusoid's frequency from that of the bin centre. This was necessary since *Wavethresh* used a non zero-padded FFT. This produces large variations in energy localisation around a sinusoidal peak for different deviations of the mean frequency from that of the centre of the analysis bin. For the sinusoidal analysis employed here a zero-padding factor of 8 is used which significantly reduces the variation in energy localisation. In fact the number of bins that require zeroing in order to produce a desired level of attenuation does not change as a function of the distance of a component from the centre frequency of an analysis bin (for example 30 bins either side of, and including, the sinusoidal peak require zero-ing to achieve 48 dB of attenuation for an 8 times zero-padded 1025 sample frame, regardless of frequency).

Since the spectral data is available in zero-padded form there are two approaches that can be taken to obtain a time domain version of the residual: decimation in the frequency domain or in the time domain. Following inverse transformation decimation in time is performed by discarding samples beyond the time support of the analysis window. Since the spectral subtraction process can spread some of the remaining component energy outside the support of the analysis window this also helps to reduce the sinusoidal energy in the residual signal. The disadvantage of not decimating before transformation to the time domain is the increased cost of the IFFT. The time domain decimation method is used here since this greatly simplifies the spectral subtraction process and offers much greater consistency in the relationship between the number of bins that are zeroed and the attenuation of deterministic components.

Non-stationarity must also be accounted for in the spectral subtraction process. Frequency non-stationarity causes a widening of the main lobe but there is little change in the energy contained in distant bins. There is no analytic method for expressing a window's power spectrum where there is frequency non-stationarity. However, the number of bins that need to be zeroed for a given level of attenuation for a particular intra-frame frequency change can be reasonably well modelled by a second order polynomial as shown in Figure 3. This illustrates the number of bins, actual and predicted, that need to be zeroed to produce an attenuation of 48 dB for a given frequency change.

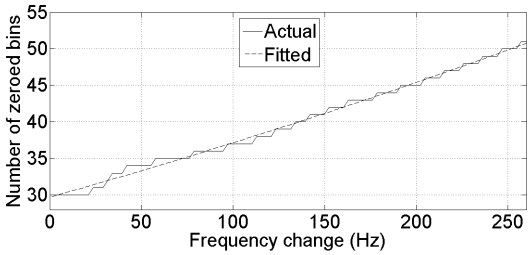


Figure 3: *Number of bins zeroed either side of peak to produce an attenuation of -48 dB for a non-stationary sinusoid versus amount of intra-frame frequency change.*

Amplitude non-stationarity can produce a significant de-localisation in the Fourier domain of a sinusoidal component. This is due to the localisation in the time domain that is produced by the amplitude change; the greater the amplitude change, the more impulse-like the component becomes. The more impulsive a component becomes the less energy it contains compared to a stationary sinusoid with the same peak amplitude. A positive amplitude change localises energy at the end of the frame and negative change localises energy at the beginning of a frame. These are the parts of the frame that experience the greatest attenuation when a window is applied.
The lower energy in a component with non-stationary amplitude combined with the attenuation introduced by the windowing process offsets the energy spreading in the Fourier domain: although zero-

ing a given number of bins produces less attenuation for a component with non stationary amplitude this loss of attenuation is compensated. This is illustrated in Figure 4 which shows the attenuation produced by spectral zeroing of 60 bins and the attenuation produced by the amplitude non-stationarity. It can be seen that the combined attenuation actually falls as the amplitude change increases. For this reason the intra-frame amplitude change for a sinusoidal component is not considered in the spectral subtraction process.

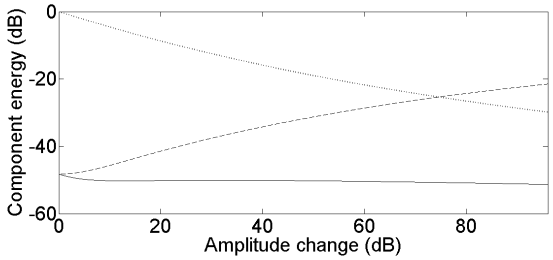


Figure 4: *Maximum component energy for a given amplitude due to intra-frame amplitude change (dotted line), energy reduction due to spectral subtraction (dashed) and combined attenuation (solid)*

4. COMPLEX B-SPLINE WAVELET ANALYSIS

Once the spectrum of the residual has been obtained via the subtraction process described in the previous section its analytic time domain version is computed. First the Hilbert transform is performed in the Fourier domain by

$$X_{\text{analytic}}(k) = \begin{cases} X(k), k = 0, \frac{N}{2} \\ 2X(k), 1 \le k \le \left(\frac{N}{2}\right) - 1 \\ 0, \left(\frac{N}{2}\right) + 1 \le k \le N - 1 \end{cases} \quad (10)$$

where N is the zero-padded transform size. The inverse transform is then computed and the output truncated so that it is the same length as the input frame. Then the B-spline wavelet transform is applied separately to the real and imaginary parts of the analytic signal. The reasons for this approach are twofold. Firstly, a major advantage of the B-spline wavelets is their compact support (both for computational speed and because only short-frames are being analysed). By having two parts of the analytic signal of the same length and analysing these separately with the B-spline filters a saving, in terms of the convolutional demands, is made over analysing the same real signal with two different transforms, one of whose filters would not be compactly supported. Secondly, since the data is already in the Fourier domain the Hilbert transform can be easily implemented at very little additional cost.

When the wavelet transform is considered as a multiresolution analysis (MRA) the sampled sequence which forms the input to the transform is considered to be the approximation of the underlying continuous signal at scale 0. However, this sequence is not the equivalent of projection of the continuous function (in this case band-limited by the anti-aliasing filter) on to the vector subspace that this scale represents. Projection is achieved by convolution of the input with a filter that is the inner product of the sinc function and the dual scaling function (the scaling function itself if the transform is orthogonal rather than biorthogonal) [11]. For the B-spline case this is achieved by convolution of the input with the sampled B-spline of the same order as that of the B-spline transform to be applied [4].

As discussed in Section 2 the B-spline wavelet approximates a Gabor function. The centre frequency of the wavelet is given by

$$f_{\text{centre},k} = \frac{f_0 F_s}{2^{k-1}} \quad (11)$$

where k is the analysis scale and $f_0 = 0.4092$ [5]. However it has been found here that (11) fails at scale 1 and that correct initialisation is only achieved by multiplication in the Fourier domain of the input with

$$F(\omega) = \text{sinc}^{m+1}\left(\frac{\omega}{2}\right) \quad (12)$$

which is the Fourier transform of the continuous, rather than sampled, B-spline of order m. If (12) is implemented in the Fourier domain then (11) holds at all scales including 1 and since the data is already in the Fourier domain there is no more expense in this filtering operation, despite the far fewer coefficients of the sampled B-spline filter in the time domain. Figure 5 shows the frequency response of the wavelet at scale 1 for both initialisation filters for a cubic B-spline. In this figure the shape given by the filter calculated from (12) is visually indistinguishable from the Gaussian function it approximates.

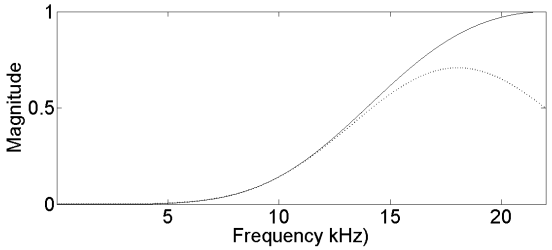


Figure 5: *Normalised magnitude response of the cubic B-spline wavelet at scale 1 for initialisation of the input sequence by the sampled (solid line) and continuous (dotted) cubic B-splines. The sample rate of the input sequence is 44.1 kHz.*

With this improved initialisation a multiresolution analysis is achieved which is akin to an atomic decomposition with Gabor functions which are successively dilated by a factor of 2. Since the critically sampled decomposition of Mallat is achieved by decimation of coefficients at each scale, aliasing is present and the transform is shift-variant [12]. A shift-invariant, non-aliased alternative to the decimated transform is the 'algorithme à trous' (algorithm with holes). This algorithm achieves dilation of the underlying wavelet and scaling functions by inserting a zero (placing a hole) in between each sample of the filters, rather than by decimating their outputs, at each successive scale [13]. Figure 6 shows the time domain, and Figure 7 the Fourier domain, shape of the wavelet filters' impulse responses at the first five analysis scales for a 1025 sample frame. Some spreading of the response in Figure 6 can be observed

at scale 1, this is an unavoidable artefact of the Hilbert transform due to its inherent band-limiting of the signal.

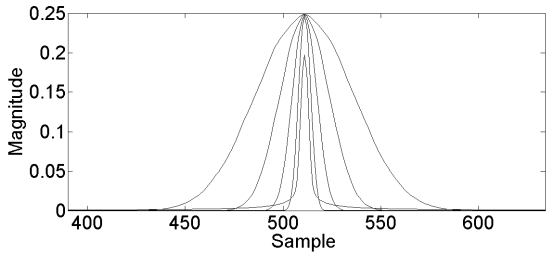


Figure 6: *The undecimated time-domain magnitude response of the complex cubic B-spline wavelet for an impulse at the centre of a 1025 sample frame. The responses widen with increasing scale.*

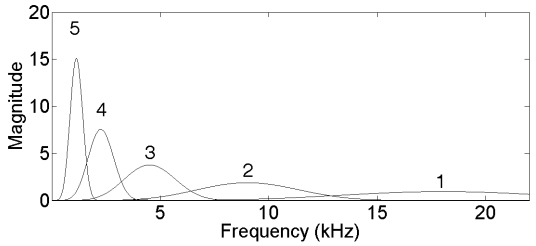


Figure 7: *Magnitude responses derived from Figure 6. The sample rate for this and subsequent plots is 44.1 kHz.*

As is the case for the DFT, the mean instantaneous frequency of a spectral component can be estimated using the complex wavelet transform, particularly since the wavelet used so closely approximates a windowed sinusoid. Reassignment is used for frequency estimation for the prior sinusoidal analysis in this system since an estimate can be obtained from a single analysis frame [10]. Unlike the STFT the wavelet transform provides more than one coefficient at each scale (apart from the highest scale of a critically sampled wavelet transform). This implies that the mean instantaneous frequency can be estimated from the first order difference of the phase between consecutive coefficients in a given scale within a single frame:

$$\overline{f}_{k,n,n+1} = \left(\phi_{\text{detail}\,k,n+1} - \phi_{\text{detail}\,k,n}\right)\frac{F_s}{2\pi} \qquad (13)$$

for an undecimated transform, where n is the index at the scale k, and ϕ is the phase, of the coefficients, and

$$\overline{f}_{k,n,n+1} = \left(\phi_{\text{detail}\,k,n+1} - \phi_{\text{detail}\,k,n}\right)\frac{F_s}{2^k\pi} \qquad (14)$$

for a decimated transform. The kth power of 2 in (14) is present since the temporal distance between indices is doubled for each increment in scale. Whilst (14) is effective for the lower half of the frequency band occupied by each scale, a correction must be applied to prevent negative frequency estimates in the upper half:

$$\overline{f}_{\text{corrected}} = \begin{cases} \dfrac{F_s}{2^k} + \overline{f}_{\text{estimated}}, & \overline{f}_{\text{estimated}} < 0 \\ \overline{f}_{\text{estimated}}, & \overline{f}_{\text{estimated}} \geq 0 \end{cases} \qquad (15)$$

An additional problem when using the decimated transform is aliasing caused by high energy in nearby out-of-scale components. A straightforward solution to this problem is to not decimate the output detail coefficients at each scale. Whilst this doubles the number of coefficients produced at each scale it does not increase the computational burden since the detail coefficients are not used in further iterations of the decimated algorithm, it is only the approximation coefficients that are used recursively. This prevents aliasing at scale 1 however aliasing still occurs at higher scales since the number of detail coefficients at each scale is reduced by decimation of the approximation coefficients at the previous scale. The ideal solution is to use the undecimated transform however this comes at a significantly increased cost than its decimated counterpart.

Circular convolution is not desirable for time-scale analysis since the purpose is to describe where events occur in (linear) time. In this analysis system the synthesis frame width is determined by the frame overlap. If frames overlap then encroachments due to circular convolution near frame boundaries can be ignored; the greater the overlap factor, the more samples that can be ignored near the boundaries. For example, with an overlap factor of 4 and a frame size of 1025 the wavelet coefficients of concern correspond to the middle 257 samples of the frame. However where circular convolution is employed a component is likely at higher scales, where the filter response is longer, to wrap into the region of interest. Therefore, for short-time wavelet analysis, circular time within frames, as opposed to linear time, makes matching of components at synthesis frame boundaries difficult. Although linear convolution is more expensive than its circular counterpart, since it increases the length of the output of each scale and, therefore, the input to the next scale, it is better suited to this application.

5. TRANSFORM COST AND PARTIAL DECIMATION

The costs of the decimated and undecimated transforms are now considered in terms of the number of multiply and add operations for the linear convolution case. For the *m*th order spline wavelet the length of the low and high pass filters in samples, at scale 1 are given by:

$$L_{\text{LPF}} = m + 2 \qquad (16)$$

$$L_{\text{HPF}} = 3m + 2 \qquad (17)$$

When a sequence of length S is convolved with a filter of length L the length of the output is $S + L - 1$. For the undecimated transform the input sequence at one scale is the approximation of the previous scale which is achieved by convolution with the dilated low pass filter. Therefore, for the undecimated transform, the sequence length before low pass filtering at scale k is given by:

$$N_k = \left(N + \left(L_{LPF} - 1\right)\sum_{n=1}^{k-1} 2^{n-1}\right) = N + \left(L_{LPF} - 1\right)\left(2^{k-1} - 1\right) \qquad (18)$$

where N is the analysis frame length. This gives a total cost for the transform of:

$$C = \left(L_{LPF} + L_{HPF}\right)\sum_{k=1}^{K} N_k \qquad (19)$$
$$= \left(L_{LPF} + L_{HPF}\right)\left(NK + \left((L_{LPF} - 1)(2^K - 1 - K)\right)\right)$$

where K is the total number of scales. For the decimated transform the filter output is decimated at each scale and so the sequence length at scale k, before filtering and decimation, is given by:

$$N_k = \left\lceil N2^{-(k-1)} + (L_{LPF} - 1)\left(1 - 2^{-(k-1)}\right)\right\rceil \\ = \left\lceil 2^{-(k-1)}(N - L_{LPF} + 1) + L_{LPF} - 1\right\rceil \quad (20)$$

Allowing for rounding up of numbers of coefficients when an odd length sequence is decimated the approximate total cost is given by:

$$C = (L_{LPF} + L_{HPF})\sum_{k=1}^{K} N_k \\ \approx (L_{LPF} + L_{HPF})\left(K(L_{LPF} - 1) + (N - L_{LPF} + 1)\left(2 - 2^{-(K-1)}\right)\right) \quad (21)$$

In order to offer some mediation between these two extremes the partially decimated wavelet transform is proposed here. The principle is straightforward: the algorithm begins by filtering the signal and inserting holes into the filter until a given decomposition level (scale) is reached, at which point the filter remains the same and the output is decimated for subsequent iterations. The only other wavelet analysis that combines decimated and undecimated transforms in this way is the over complete DWT (OCDWT) described in [14]. However, this system begins with decimation and then at higher scales switches to filter dilation. This order is reversed in the system proposed here since this reduces shift variance at all scales.

Equations (18) – (21) can be combined to calculate the cost of the partially decimated transform. The cost of calculating the undecimated scale coefficients can be calculated directly from (19) where N is the length of the input sequence and $K = U$ is the number of undecimated scales. The cost of calculating the subsequent decimated coefficients is given by a modified version of (21):

$$C = (L_{LPF} + L_{HPF})\sum_{d=1}^{D} N_d \\ = (L_{LPF} + L_{HPF})\left(D(L_{dLPF} - 1) + (N_{undec} - L_{dLPF} + 1)\left(2 - 2^{-(D-1)}\right)\right) \quad (22)$$

where D is the number of decimated scales and N_{undec} is the length of the final approximation sequence output from the undecimated part of the transform, given by (18) where $k = U + 1$, which is then halved (since this sequence is decimated before the next filtering stage). L_{dLPF} is the length of the dilated LPF and is given by

$$L_{dLPF} = (L_{LPF} - 1)\,2^{U-1} + 1 \quad (23)$$

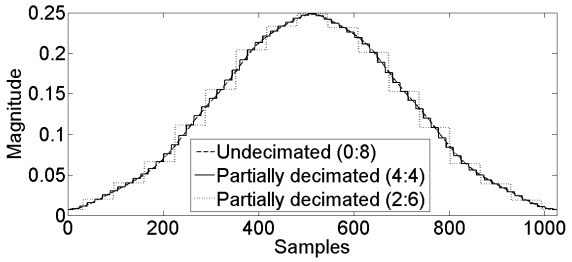


Figure 8: *Time-domain magnitude response at scale 8 of the complex cubic B-spline wavelet transform for differing amounts of decimation.*

Figure 8 shows the time domain magnitude response at scale 8 for an impulse in the centre of the analysis frame for different ratios of numbers of decimated to undecimated scales.

6. SPLIT WAVELETS FOR BANDWIDTH ESTIMATION

The frequency-splitting 'trick' described in [15], and used to produce the full binary tree decomposition used in wavelet packets, is used here to produce estimates of the bandwidth of components at each scale. At one extreme, the instantaneous mean centre frequencies of the scale filter and the two split filters will coincide for an impulse in the frequency domain and, at the other, their centre frequencies will be the same as those of the fixed filters for an impulse in the time domain. Therefore the proximity of the derived centre frequencies for the two complex split filters can be used to estimate the width of the underlying component.

For the undecimated transform the split at each scale is achieved by filtering of the detail coefficients at that scale. The filters are obtained by dilation by a factor of two of the high and low pass filters used to derive the approximation and detail coefficients. For the decimated transform the split can be achieved by convolution of the decimated detail coefficients with the existing filters. However this would produce fewer split than scale coefficients meaning that there could not be a one-to-one mapping of a scale coefficient to its lower and upper split coefficients. Therefore, in the split implementation described here, the filters are dilated and the scale coefficients left undecimated whether the split is occurring for a decimated or undecimated scale in the partially decimated transform. It should be noted that where splitting *is* performed then not decimating the detail coefficients at each scale (discussed earlier as a method of reducing aliasing) will increase the computational cost.

Splitting at a given scale is achieved by convolution of the detail signal with the low and high pass wavelet filters dilated by a factor of two from those used to generate the approximation and detail coefficients at that scale. Dilation of a filter's impulse response in the time domain is equivalent to an equal contraction of its response in the frequency domain. Therefore the frequency responses of the split wavelets' filters are given by:

$$\Psi_{\text{lower}}(\omega) = \Psi_{\text{scale}}(\omega)\,HPF_{\text{scale}}(2\omega) \quad (24)$$

$$\Psi_{\text{upper}}(\omega) = \Psi_{\text{scale}}(\omega)\,LPF_{\text{scale}}(2\omega) \quad (25)$$

where Ψ and XPF are the Fourier transforms of the various filters. The perhaps counter-intuitive result that the upper split wavelet is produced by convolution with the LPF and the lower split by convolution with the HPF is explained by the fact that it is the aliased (reflected) parts of the filters' frequency responses (which are contracted by a factor of 2 in the above equations) that coincide with the region where the response of the wavelet filter is greatest. The upper part of Figure 9 shows the magnitude frequency responses of the wavelet and its upper and lower splits at scale 1. The lower part of this figure shows the shape of the underlying continuous functions. As would be expected of the dilation and convolution operations of the splitting operations, the split wavelets have greater time support but are more localised in frequency than the parent wavelet.

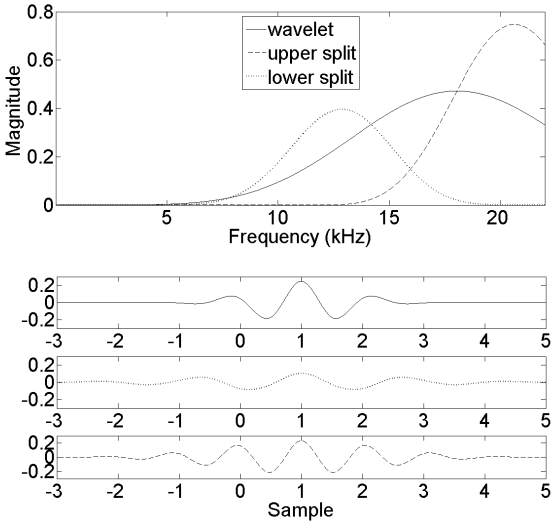


Figure 9: *Magnitude frequency response (top) and time domain shape of cubic B-spline wavelet and its splits at scale 1.*

The centre frequencies of the split wavelets at each scale are given by (11) where f_0 = 0.2919 for the lower and 0.4678 for the upper splits respectively [8]. Therefore the maximum difference (i.e. that due to an impulse) between split filters at scale *k* is given by:

$$\Delta f = \frac{0.1760 F_s}{2^{k-1}} \tag{26}$$

Figure 10 illustrates how differences between frequency estimates at a single scale occur where a component has spectral breadth. The frequency estimates at scale 1 for the wavelet and its splits are shown for a sinusoid and for a single impulse which occurs in the middle of the frame (sample 513). There is a clearly visible difference in estimates for the impulse whereas, at the same scaling of the vertical axis, there is no difference in estimates for a stationary sinusoid.

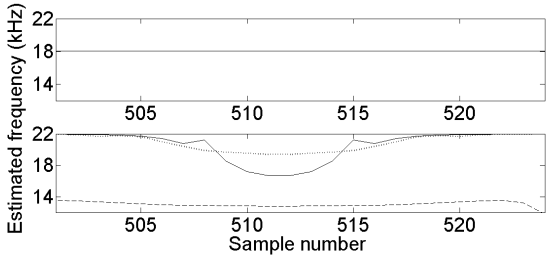


Figure 10: *Frequency estimates for a sinusoid (top) and an impulse at the centre of the frame (bottom).*

The cost of the split transform is the cost of the non-split transform, given by equations (18) to (23), plus the cost of filtering that produces the splits at each scale. The split at each scale is achieved by high pass filtering of the detail coefficients at that scale followed by high and low pass filtering with filters which are dilated by a factor of two from those used to produce the approximations and details at that scale. For the undecimated transform the sequence length, Ns_k (the s indicates 'split'), before the high and low pass split filtering is given by:

$$Ns_k = \left(N + \left(L_{LPF} - 1 \right) \sum_{n=1}^{k-1} 2^{n-1} \right) = N_k + 2^k \left(L_{HPF} - 1 \right) \tag{27}$$

and so the combined cost of the all the splitting stages is given by:

$$\begin{aligned} Cs &= \left(L_{LPF} + L_{HPF} \right) \sum_{k=1}^{K} Ns_j = \left(L_{LPF} + L_{HPF} \right) \sum_{k=1}^{K} N_j + 2^{k-1} \left(L_{HPF} - 1 \right) \\ &= \left(L_{LPF} + L_{HPF} \right) \left(NK + \left((L_{LPF} - 1)(2^k - 1 - K) \right) + \left(L_{HPF} - 1 \right) \sum_{k=1}^{K} 2^{k-1} \right) \\ &= \left(L_{LPF} + L_{HPF} \right) \left(NK + \left((L_{LPF} - 1)(2^K - 1 - K) \right) + \left(L_{HPF} - 1 \right) \left(2^K - 1 \right) \right) \end{aligned} \tag{28}$$

and the total cost of the transform is given by adding (19) and (28). For the decimated transform the sequence prior to splitting is the detail sequence at that scale. This is given by

$$Ns_k = \left\lceil 2^{-(k-1)} \left(N - L_{LPF} + 1 \right) + L_{LPF} + L_{HPF} - 2 \right\rceil \tag{29}$$

and the total cost of the splitting stage is given by adapting (21):

$$C = \left(L_{LPF} + L_{HPF} \right) \left(K \left(L_{LPF} - 1 \right) + \left(N - L_{LPF} + 1 \right) \left(2 - 2^{-(K-1)} \right) + J \left(L_{HPF} - 1 \right) \right) \tag{30}$$

The total cost of the split decimated transform is given by adding (21) and (30). The cost of the splitting stage of the partially decimated transform can be calculated for the undecimated levels by the same sum. The cost of splitting at the decimated levels is given by a modification of (30)

$$\begin{aligned} C &= \left(L_{LPF} + L_{HPF} \right) \\ &\times \left(K \left(L_{dLPF} - 1 \right) + \left(N_{undec} - L_{dLPF} + 1 \right) \left(2 - 2^{-(K-1)} \right) + K \left(L_{dHPF} - 1 \right) \right) \end{aligned} \tag{31}$$

where L_{dHPF} is given by

$$L_{dHPF} = \left(L_{HPF} - 1 \right) 2^{U-1} + 1 \tag{32}$$

Finally, the total cost of the partially decimated split wavelet transform is given by adding (22) and (32). Figure 11 shows the computational cost of the split and un-split, decimated and undecimated complex cubic spline wavelet transforms for linear convolution. For comparison the cost of a 1024 and 8192 point FFT are also shown.

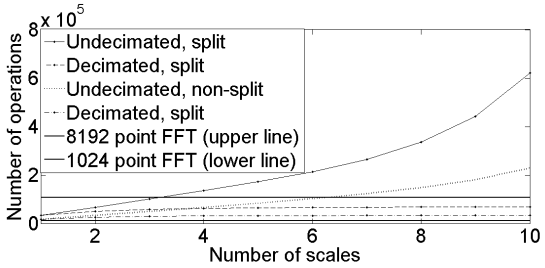


Figure 11: *Number of complex multiply and odd operations required for various complex cubic B-spline wavelet transforms of a 1025 sample frame.*

7. APPLICATION

The complex wavelet analysis system described is able to adapt to different types of input component. The frame-by-frame spectral modelling system in which it is used employs biquadratic parametric equalisers applied to a white noise source for resynthesis of the residual. Two examples are now given which demonstrate how this system performs on different types of input signal. The top part pf Figure 12 shows a resynthesized sequence of unity impulses. In this case the time localisation is good, with energy focussed in a small number of samples. At the other extreme the bottom part shows the time domain input, output and magnitude frequency response of a stationary sinusoid. Although such a component is unlikely to form part of the residual, it demonstrates the ability of the resynthesis filters to adapt their bandwidth to give good frequency localisation and to shift their centre frequency to that of the input component. This time-frequency adaptation is made possible by the bandwidth estimation described in this section.

Figure 13 demonstrates how the residual synthesis can adapt in a single frame. The time localisation at the onset is good but this changes to good frequency localisation later on in the frame (the analysis overlap factor is 2 so the synthesis frame is half the size of the analysis frame). During the last half of the frame the sinsusoidal oscillator ramps on exponentially, 'taking over' from the residual synthesis by the next frame.

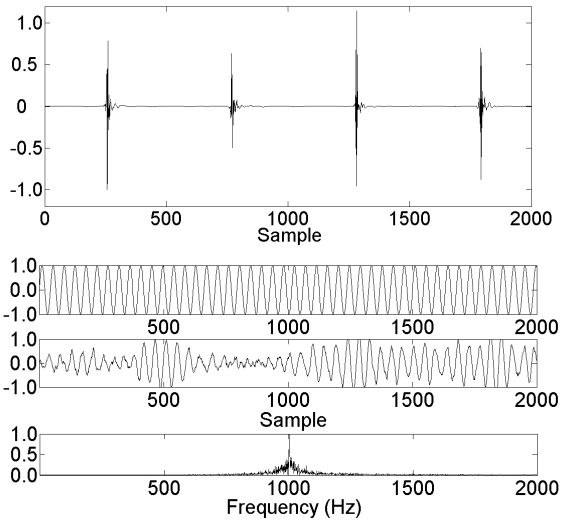


Figure 12: *Residual resynthesis of time domain (top) and frequency domain impulses (bottom).*

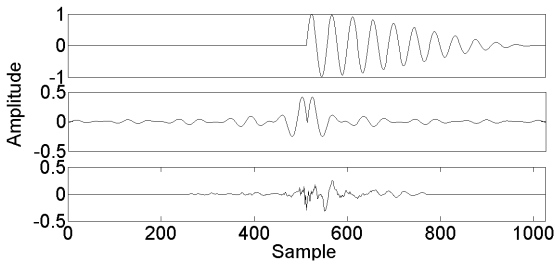


Figure 13: *Windowed sinusoid with sudden onset (top), residual after spectral subtraction (middle) and resynthesized residual (bottom).*

8. SUMMARY

A multiresolution analysis system which produces estimates of magnitude, mean instantaneous frequency and bandwidth of components, and is suited to a residual modelling system has been presented and placed in the context of a frame-by-frame spectral modelling system. The properties of the wavelet analysis, its cost, and partial decimation as a means of negotiating between computational cost and shift-variance\aliasing have been described. Further work will look at how the synthesis and analysis filters can be better matched whilst retaining the simplicity of the resynthesis method. A more detailed treatment and analysis of the work presented here (including measures of aliasing and shift-invariance for different levels of partial decimation) can be found in [8].

9. REFERENCES

[1] X.Serra, "A System for Sound Analysis/Transformation/Synthesis Based on a Deterministic plus Stochastic Decomposition", PhD thesis, Stanford University, USA, 1989.

[2] T. Verma and T. Meng, "An Analysis\Synthesis Tool for Transient Signals", *Computer Music Journal*, vol. 24, pp. 47-59, Summer 2000.

[3] M. Unser, "Splines, A Perfect Fit for Signal and Image Processing", *IEEE Signal Processing Magazine*, pp. 22-38, November 1999.

[4] M. Unser et al, "A Family of Polynomial Spline Wavelet Transforms", *Signal Processing*, vol. 30, pp. 141-162, January 1993.

[5] M. Unser et al, "On the Asymptotic Convergence of B-Spline Wavelets to Gabor Functions", *IEEE Trans. on Information Theory*, vol. 38, pp. 864-872, March 1992.

[6] C. Chui and J. Wang, "On Compactly Supported Spline Wavelets and a Duality Principle", *Trans. of the American Mathematical Society*, vol. 330, pp. 903-915, April 1992.

[7] J. Wells and D. Murphy, "High-Accuracy Frame-by-Frame Non-Stationary Sinusoidal Modelling", *Proc. of the Int. Conf. on Digital Audio Effects (DAFx-06)*, pp.253-258, 2006.

[8] J. Wells, "Real-Time Spectral Modelling of Audio for Creative Sound Transformation", PhD Thesis, University of York, January 2006, Available at http://www.jezwells.org

[9] X. Amatriain, "Spectral Processing", chapter in (ed. Zölzer), *DAFX – Digital Audio Effects*, John Wiley, Chichester, 2002.

[10] J. Wells and D. Murphy, "Real-Time Spectral Expansion for Creative and Remedial Sound Transformation", *Proc. of the Int. Conf. on Digital Audio Effects (DAFx-03)*, pp. 61-64, 2003.

[11] P. Abry and P. Flandrin, "On the Initialization of the Discrete Wavelet Transform", *IEEE Signal Processing Letters*, vol. 1, pp. 32-34, February 1994.

[12] S. Mallat, "A Wavelet Tour of Signal Processing", Academic Press, San Diego, 1999.

[13] M. Shensa, "The Discrete Wavelet Transform: Wedding the À Trous and Mallat Algorithms", *IEEE Trans. on Sig. Proc.*, vol. 40, pp. 2464-2482, October 1992.

[14] A. Bradley, "Shift-Invariance in the Discrete Wavelet Transform", *Proc. of the 7th Conf. on Digital Image Computing: Techniques and Applications*, pp. 29-38, December 2003.

[15] I. Daubechies, "Ten Lectures on Wavelets", Society for Industrial and Applied Mathematics, Philadelphia, 1992.

A COMPLEX ENVELOPE SINUSOIDAL MODEL FOR AUDIO CODING

Maciej Bartkowiak

Chair of Multimedia Telecommunications and Microelectronics,
Poznań University of Technology
Poznań, Poland
mbartkow@multimedia.edu.pl

ABSTRACT

A modification to the hybrid sinusoidal model is proposed for the purpose of high-quality audio coding. In our proposal the amplitude envelope of each harmonic partial is modeled by a narrowband complex signal. Such representation incorporates most of the signal energy associated with sinusoidal components, including that related to frequency estimation and quantization errors. It also takes into account the natural width of each spectral line. The advantages of such model extension are a more straightforward and robust representation of the deterministic component and a clean stochastic residual without ghost sinusoids. The reconstructed signal is virtually free from harmonic artifacts and more natural sounding. We propose to encode the complex envelopes by the means of MCLT transform coefficients with coefficient interleave across partials within an MPEG-like coding scheme. We show some experimental results with high compression efficiency achieved.

1. INTRODUCTION

Parametric audio coding [1] is usually considered as a departure from the waveform coding paradigm in a sense that matching of absolute signal value is abandoned in favor of matching perceptually relevant features. Parametric approach promised an exciting perspective of data reduction almost down to the amount of semantic content, thus offering an option for great coding efficiency. The problem is that such extreme compression requires very flexible and realistic models, at least for those signal features that are essential from perception point of view. This goal remains elusive in current implementations which have yet to prove their advantage over latest transform coding techniques, such as MPEG-4 HE-AACv2 [2,3].

In fact, the borders between parametric and waveform coding are quite blurred. Current perceptual codecs often feature parametric enhancements to the traditional transform-based schemes. Parametric tools like PNS (Perceptual Noise Substitution), SBR (Spectral Band Replication) and PS (Parametric Stereo) helped to push the limits of transform coding down to the range of 24-32kb/s while still offering a good quality of reconstructed audio. Therefore it is reasonable to consider MPEG-4 HE-AACv2 as a hybrid transform-parametric technique.

Purely parametric coding of wideband audio traditionally employs a well established hybrid model to represent the main spectral features of the signal in terms of deterministic and stochastic components. The deterministic component is modeled as a sum of non-stationary sinusoids,

$$\hat{s}(t) = \sum_{k=1}^{N} A_k(t) \cos\left(\varphi_k + 2\pi \int_0^t f_k(\tau)\, d\tau \right), \qquad (1)$$

as proposed by McAulay and Quatieri [4] and improved later by others, e.g. [5,6]. It is generally assumed that the magnitudes and frequencies of constituent sinusoids evolve slowly in time and they may be very well approximated by simple functions. For example, $A_k(t)$ is usually a piecewise linear ramp and $f_k(t)$ is a low order polynomial. The stochastic part is usually considered as a residual obtained during an analysis by synthesis process, after spectral subtracting the estimated sinusoidal part from the original signal, as proposed by Serra [7] and further refined, e.g. [8,9]. The stochastic part is usually modeled by filtered noise with an additional envelope (2)

$$\hat{n}(t) = A_n(t)\left[h_n(t) * \varepsilon(t)\right], \quad \varepsilon \propto \boldsymbol{N}(\mu,\sigma), \qquad (2)$$

where $\varepsilon(t)$ represents a white noise process, and $h_n(t)$ represents the impulse response of an AR or ARMA modeling filter [10]. Some more elaborate models feature additional functions for efficient representation of transients, e.g. [11,12,13]. These are usually detected and removed from the original signal at the beginning of the analysis by synthesis process.

There are several successful applications of the above hybrid model to compression of wideband audio with the most important being the one covered by ISO/MPEG-4 SSC standard [13,14]. Although the codec implementation available from ISO shows a great compression efficiency, it is unable to offer a truly high quality output, and many listeners complain on unnatural sounding harmonic clashes that are particularly audible in sounds rich with overtones (*glockenspiel, trumpet*) and human voice (famous *Suzan Vega* sample). Since about 80% of the total bit stream produced by the encoder is used for the sinusoidal part, we consider some serious deficiency of the underlying model to be responsible for these artefacts.

2. DRAWBACKS OF THE SINUSOIDAL MODEL

There is a lot of research on the sinusoidal model alone. The most important problem is an accurate estimation of the parameters (e.g. [13,14]) such that the reconstructed sum of time-varying sinusoids (1) matches the tonal part of the signal as closely as possible for the analysis by synthesis principle to work in time domain. This in general is difficult if the tonal part is non-stationary or buried in noise. Apart from well-known time/frequency resolution limits due to the analysis window length and shape, there is a bias related to AM and FM components [15,16,17], and the estimation accuracy is constrained by the Cramer-Rao bound.

First of all, inaccurate estimation of frequency and amplitude for each partial leads to bulk of the tonal energy being left in the residual signal (fig. 1). These so called "ghost sinusoids" are a significant source of inaccuracy of the low-order auto-regressive model being fitted to the residual PSD. On the other hand, if the

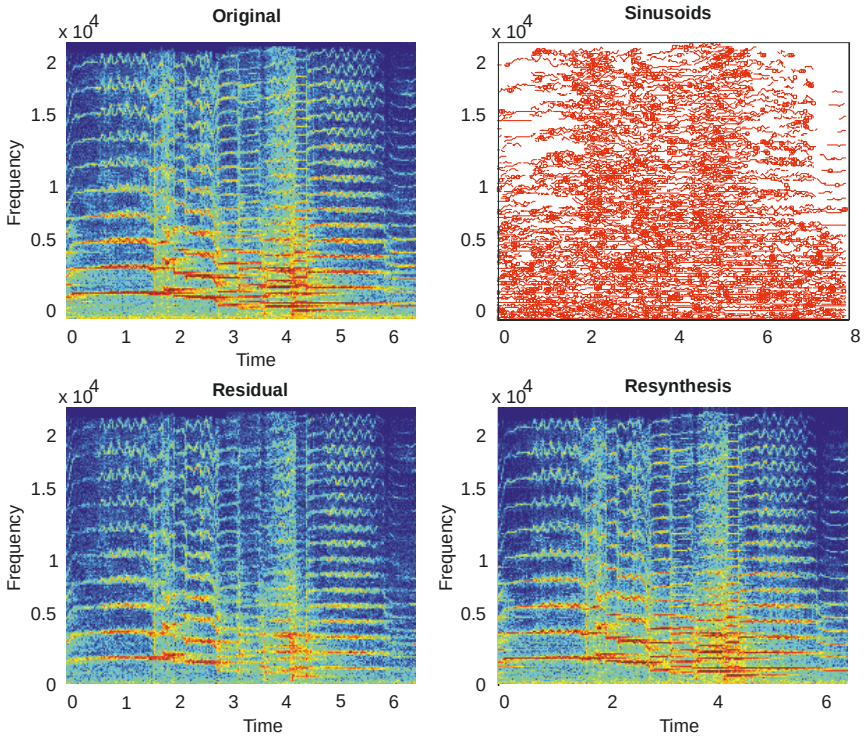


Figure 1: *Sinusoidal plus noise analysis demonstrating limitations of the sinusoidal model*

sinusoids are estimated and extracted from the original signal one by one, there is a whole bulk of sinusoids representing each of the individual tonal partials, and the model is simply inefficient. Both problems have been addressed with some successful solutions [18,19,20], however perfect results are obtained only for very stationary sounds or artificial spectra. In case of real audio signals, small random fluctuations of amplitudes and frequencies observed on short-time spectrograms of natural sounds are not very well represented by the traditionally formulated model. Furthermore, parameter quantization [13,21] which is an essential component of every compression technique introduces small discrepancies into the encoded frequencies, usually up to ±0.5% [13]. Frequency deviation of 0.088 ERB is generally considered as imperceptible with regard to single tones or fused harmonics heard in isolation. However, it is not so in case of several components of harmonic series beating against each other due to different frequency quantization error. In such case, small offsets destroy fixed phase relationships between overtones and cause a sensation of mistuning and unnaturalness.

In our opinion, the classic sinusoidal model (1) exhibits two significant drawbacks when considered as a compression tool:

1. it is too sensitive to small inaccuracies of parameter estimation and representation, since even little frequency errors lead to significant modeling problems or even audible artifacts,
2. it is too idealistic, since it assumes an infinitely small instantaneous bandwidth of each sinusoidal partial, while in real audio signals the tonal components exhibit a significant spectral width.

The basic idea behind the extension of the sinusoidal model proposed in this paper is to incorporate the narrowband content associated with each partial into its amplitude envelope. Instead of a piecewise linear functions, the envelopes $A_k(t)$ are modeled as LF signals which are heterodyned to proper frequency by corresponding complex sinusoidal carriers. Since the amplitudes are band limited complex signals, they may be represented with significantly reduced sampling rate and using one of the well established signal coding techniques, in our case – transform coding.

Fitz and Haken proposed bandwidth-enhanced sinusoids [22] obtained through narrowband frequency modulation with a filtered noise modulator as a flexible tool for modeling the stochastic component of the signal. In the context of encoding the deterministic part, this enhanced model is not applicable since the representation does not guarantee waveform matching. While bandwidth enhanced sinusoids offer easy parameterization of a narrowband stochastic process, our complex amplitude model is a more systematic expression of the signal deterministic content that allows for near transparent quality at sufficiently high data rate.

3. PROPERTIES OF THE COMPLEX ENVELOPE

Every narrowband signal may be expressed as a product of modulation of a low-frequency band-limited "content" (the complex envelope) by a complex sinusoidal carrier (3). We use this expansion to represent the constituent partials of the sinusoidal model.

$$s_k(t) = \mathrm{Re}\left\{ x_k(t)\, \mathrm{e}^{j 2\pi f_k t} \right\} \tag{3}$$

In order to study the spectral properties of the envelope, let us consider an example of a high violin note with vibrato (fig. 2). Due to the variations of fundamental frequency, short-time frequency analysis with a reasonable window length (here: N=2048) shows a series of thick bulges in the magnitude spectrum.

Complex amplitude envelopes may be obtained for each of the existing sinusoidal component through frequency shift according to their instantaneous frequencies. For this purpose we detect and

track the sinusoidal components of the signal using the McAulay-Quatieri algorithm. We consider only long solid tracks as carriers of tonal content in our model. After demodulation, the remaining bandwidth of each envelope is mostly related to frequency estimation errors, the fluctuation of the instantaneous frequency, and last but not least – the spectrum of the magnitude envelope of the whole sound. Experiments show that the estimated complex envelope signals are very narrowband (fig. 3) therefore they may be very efficiently encoded using transform coding with only few significant coefficients. Compared to sinusoidal coding with piecewise-linear envelope this scheme needs more data to represent several transform coefficients, however it allows for much lower update rate (long frames).

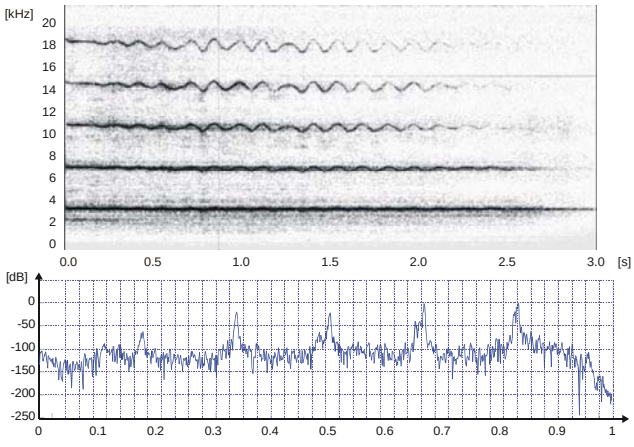


Figure 2: *A spectrogram of a violin note (above) and a corresponding STFT magnitude at t=0.8*

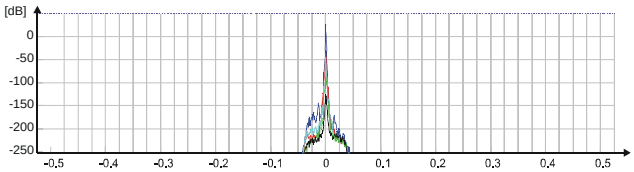


Figure 3: *PSD-s of the complex envelopes (5 partials) obtained from the example test signal (fig. 2)*

Transform coding of audio spectra is usually based on coefficients of MDCT transform. It may be shown that in case of complex-valued signals the optimal extension of this scheme is the use of modulated complex lapped transform (MCLT) proposed by Malvar [23],

$$X(r) = \sum_{n=0}^{2N-1} x(n)\, w(n)\, \mathrm{e}^{-j\frac{\pi}{N}\left(n+\frac{N+1}{2}\right)\left(r+\frac{1}{2}\right)}, \qquad (4)$$

where $x(n)$ denotes the time-domain signal, and $w(n)$ denotes a real-valued window function satisfying the conditions for aliasing cancellation as defined by Princen, Johnson and Bradley [24]. MCLT is an extension of MDCT in a sense that the real part of MCLT is equivalent to MDCT which is based on DCT-4, while the imaginary part is based on DST-4. Thus it offers a critically sampled filterbank with TDAC working for both the real and imaginary parts, and it may be implemented using FFT.

For encoding of complex envelope signals with MCLT we adopt the well established data compression scenario as specified in MP3 and AAC standards. In our implementation, the transform is followed by coefficient perceptual scaling, quantization and entropy coding. In fact, the main difference is the treatment of the complex-valued coefficients, $X(r)$.

An interesting observation from the analysis of the complex envelopes (fig. 3) is also that these signals are similar in their magnitude spectrum shape. Since harmonics having a common source (e.g. overtones of the same fundamental) have also a common magnitude envelope, a significant portion of the spectral content related to this envelope is usually present in the complex envelope signals. This suggests that an additional coding gain may be achieved in exploiting inter-partial correlation within transform coding. Our proposal consists in application of a simple coefficient interleave scheme which is applied to those sets of sinusoidal partials which are detected as being components of harmonic series. This requires an identification of harmonic series and proper grouping of the sinusoidal tracks before coding.

4. CODING TECHNIQUE

4.1. Proposed codec structure

The proposed audio codec (fig. 4) operates on the signal arranged in frames of 2048 samples with 50% overlap. The input signal is analyzed using FFT. Local maxima in the magnitude spectrum are detected, selected according to the energy of corresponding harmonic partials, and exact frequencies are estimated according to Marchand's derivative algorithm [14,16]. A tracking algorithm attempts to connect corresponding points of the frequency grid across consecutive analysis frames and thus to create the map of sinusoidal tracks. The tracks are grouped into sets corresponding to harmonic series with common fundamental frequency, and sent to the decoder.

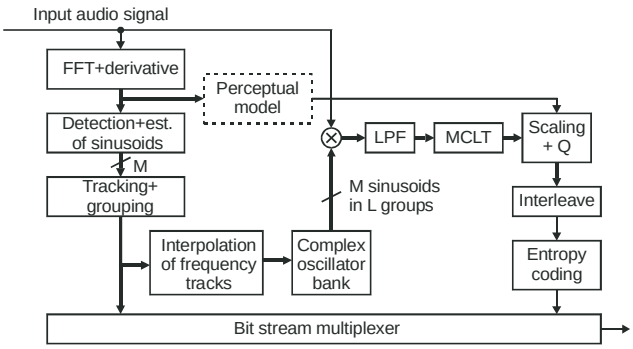


Figure 4 : *The structure of the proposed encoder*

A bank of M carrier generators (complex sinusoidal oscillators) is driven by the estimated frequencies. The original signal is independently heterodyned by each of the carriers, thus providing an effective SSB-like frequency shift towards DC. The resulting M complex signals are lowpass filtered for rejecting the unwanted products. In our implementation we use a fixed zero-phase 256-tap FIR filter with stopband attenuation of 65dB. There is a natural trade-off between the amount of side energy around each sinusoidal partial in frequency domain and the energy of the residual error. First of all, the aim is to avoid leaving any tonal energy in

the residual. Therefore the bandwidth of the filter should be determined with respect to the accuracy of the frequency estimation algorithm.

The set of complex LF envelopes is subsequently encoded in the following way. First, all signals are subject to the MCLT transform. The coefficients are appropriately scaled with application of the perceptual model, and quantized. A coefficient interleave process follows. An independent vector of coefficients is created for each of the groups of envelopes belonging to different harmonic series. In each group the coefficient vector is constructed by taking consecutive coefficients one by one from each of the partials. In other words, first coefficient from the lowest partial is followed by the first coefficient from the second partial, and so on (fig. 5). Independent vectors are constructed from the real and imaginary coefficients. These are subject to subsequent entropy coding.

4.2. Estimation, interpolation, tracking, grouping, and encoding of partial frequencies

Estimation of sinusoidal frequency based on frame analysis usually assumes that the resulting value approximates the instantaneous frequency (IF) of given partial at the middle of analysis frame. The frequency values are transmitted to the decoder once per frame and should be interpolated on a sample basis for a continuous demodulation of sinusoidal partial. This is necessary in the encoder since the aim is to obtain the complex envelopes as narrowband as possible in order to maximize the transform compression gain. It is also necessary in the decoder, in order to properly shift the reconstructed spectra back to the right place.

The problem of appropriate frequency interpolation that minimizes phase errors was studied with the development of the sinusoidal model, and a solution using cubic polynomial was proposed [4,7,13]. We basically follow this interpolation scheme, but no significant penalty has been observed by application of a simpler linear interpolation. In fact, phase matching is not necessary since the content is encoded in complex envelope. Our extended model is also quite insensitive to small frequency errors, since their only manifestation is in little increase of envelope bandwidth and transform coefficient values.

Proper operation of the codec certainly depends on reliable tracking of the frequencies of sinusoidal partials. Big tracking errors such as those occurring in case of crossing sinusoidal trajectories lead to audible artifacts (e.g. temporal discontinuities in tonal energy similar in timbre to the *flanger* effect). For robust tracking we employ a modified McAulay-Quatieri algorithm [4] with relaxed birth/death conditions and different matching criteria. Our matching technique aims at better smoothness of tracks, which is achieved by seeking for the best match among those frequency points in consecutive frame that minimize the second derivative of frequency. In our experience, such principle allows to some extent for coping with the problem of crossing tracks and deep frequency modulation.

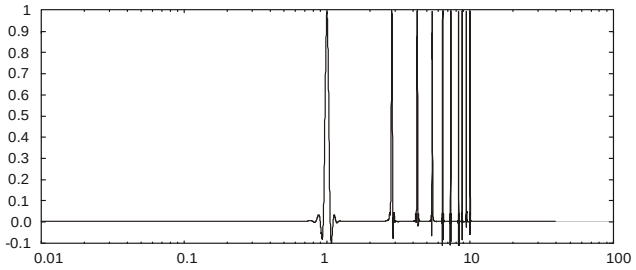


Figure 6: *The template used for detection of harmonic series*

A following procedure is employed for grouping of tracks into harmonic series. At first, candidate fundamental frequencies $\{\hat{f}_1, \hat{f}_2 \ldots \hat{f}_L\}$ are determined by correlating in frequency domain the magnitude spectrum resampled to log frequency scale with a constant-Q harmonic template (fig. 6). The idea is to exploit the property of shift in log domain being equivalent to scaling in linear domain, which is required to estimate the best matching of the

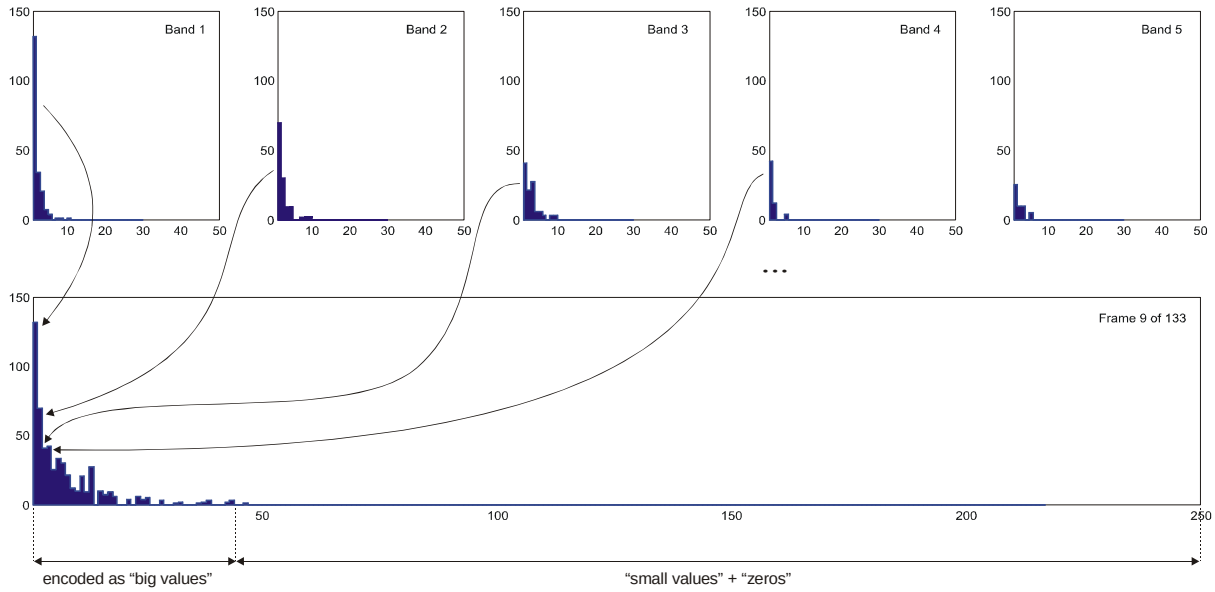


Figure 5: *Coefficient interleave within one group of partials, and coding in sections*

harmonic series to the template [25]. We use a high resolution (16384 points) log frequency representation that allows us to find the fundamental frequency using FFT-based correlation with an accuracy of about 1.37ct.

A given frequency track $f_k(t)$ is classified as belonging to one of the candidate harmonic series $\{f_1(t), 2f_1(t), 3f_1(t),..\}$, $\{f_2(t), 2f_2(t), 3f_2(t),..\}$, ... $\{m\, f_L(t), m=1,2...\}$ that minimizes

$$dist(f_k, f_l) = \frac{d}{dt} f_l(t) - \frac{f_l(t)}{f_k(t)} \frac{d}{dt} f_k(t), \quad l = 1 \ldots L \tag{5}$$

Finally, the fundamental frequencies of each harmonic series are estimated by

$$f_l = \frac{1}{\aleph_{\{m\,\hat{f}_l\}}} \sum_{f_k \in \{m\,\hat{f}_l\}} \frac{f_k}{round(f_k / \hat{f}_l)}, \qquad l = 1 \ldots L\,. \tag{6}$$

The frequencies are encoded and transmitted to the decoder in groups, using a representation that in our experience minimizes data overhead. For each group, only the fundamental frequency is represented with a natural binary code. The remaining frequencies $f_1 < f_2 < \ldots f_M$ are represented by differences between integer multiple of the fundamental f_l, and the actual value,

$$\Delta f_k = f_k - m\, f_l\,, \quad \text{where} \quad m = round(f_k / f_l)\,. \tag{3}$$

The fundamental frequency f_l and a set of differences Δf_m are quantized uniformly with quantization step equal to half of the frequency resolution of MDCT, and encoded by a dedicated Huffman code. Both encoder and decoder share identical dequantization rule.

4.3. Scaling, quantization and entropy coding of the complex envelope signals

Quantization of MCLT coefficients in all complex envelope signals is done in a very similar way to the MPEG-4 AAC algorithm. A nonlinear quantizer is used independently for the real and imaginary part, and the degree of quantization is controlled by coefficient scaling,

$$\overline{X_k[r]} = \mathrm{sgn}(X_k[r])\; floor\left[2^{(scf_k - gsf)/4} \left| X_k[r] \right|^{3/4} + 0.0946\right]. \tag{7}$$

Individual scaling factors scf_k are determined for each of the envelope signals, plus one global gain factor, *gsf* controls the degree of distortion of all partials. All coefficients of each envelope signal X_k share the same scaling factor scf_k. Such approach leads to uniform distribution of the quantization noise around each partial so that it may be masked by the energy of spectral peak. It also allows to adapt an effective bit allocation algorithm primarily developed for an AAC coder.

In fact, our coding technique is quite similar to traditional transform coding, since the coding error has a form of a narrowband noise. Therefore a perceptual model developed for the family of MPEG L3/AAC techniques is also applicable here. The only simplification is that there is no need to calculate the tonality index for the maskers, and the final masking threshold is calculated on the basis of tone-masking-noise (TMN) coefficient. The scaling factors scf_k in (7) are therefore calculated on the basis of the masking threshold determined by the perceptual model

Entropy coding of the quantized MCLT coefficients implements a typical scheme of data sectioning into "big values" and "small values" taken from the MP3 algorithm. Due to coefficient interleave, the distribution of quantized values along the data vector is concentrated near its beginning (fig. 5). For entropy coding we use a coding scheme taken literally from the MP3 technique. All the "big values" with magnitudes not exceeding 15 are encoded in pairs, using 2D codewords from selected Huffman tables. The whole section is divided into three equal groups, and an optimal Huffman table is selected for each group. Very big values are represented as escape codes. Values from the range of <-1…1> are encoded in quadruples using a dedicated Huffman table.

5. EVALUATION

In order to verify the advantages of the proposed coding technique over traditional parametric coding, a series of experiments has been carried out. First, a hybrid sinusoidal+noise model has been implemented in Matlab. A second version of the same model featuring complex envelopes and MCLT-based coding has been prepared. Both implementations share identical procedures for estimation and tracking the sinusoids, but no perceptual model is used. Both the sinusoidal parameters and the transform coefficients are quantized in a uniform way. The noise residual is modeled using a warped LPC algorithm. Instead of entropy coding, a simple entropy measure is used to estimate the amount of information contained in both representations of the signal.

A test suite consisting of several music excerpts (violin, opera voice, trumpet) has been used to compare the performance of both models. The reconstructed signals have been compared in a blind listening test with degree of quantization controlled in such a way to force the output entropy to be similar. Figure 7 shows an example reconstructed deterministic part and corresponding residual signal. These should be compared with figure 1. Figure 8 shows the subjective listening test results (mean opinion score of 7 lis-

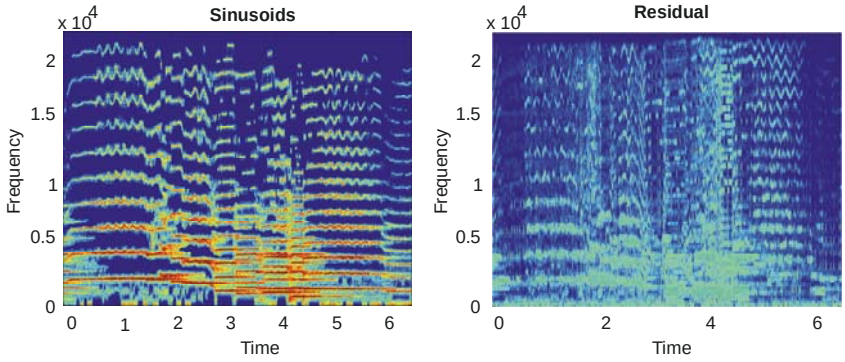


Figure 7: *Reconstructed deterministic part and noise residual after coding with complex envelope and MCLT quantization*

teners) for H=15kb/s and H=30kb/s.

The general conclusion form the first test is that there is a significant improvement of the subjective quality achieved thanks to more truthful reconstruction of the sinusoidal component of the signal. In fact, thanks to more accurate reconstruction of the deterministic part, also the noise residual is much better represented. Compared to traditional sinusoidal model, the output of our codec sounds more natural and is free from typical artifacts attributed to inappropriate sinusoidal parameters.

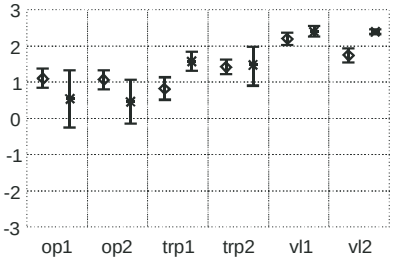


Figure 8: *Subjective test results (MOS) for 6 items in 7-point ITU scale. Positive values show a preference of the new model. Diamonds: 15kb/s, stars: 30kb/s.*

6. CONCLUSIONS

A new approach for encoding of the deterministic part within a parametric audio coder is proposed in the paper. Our extended sinusoidal model uses complex envelopes to represent the narrowband spectral content around each encoded sinusoid. This content is encoded using transform coding. The proposed scheme may be considered as a hybrid of perceptual and transform coding. It may also be interpreted as an adaptive subband coding with subbands following the instantaneous frequencies of individual harmonics in the signal. The experimental results show that a combination of this model with an advanced transform coding technique featuring coefficient interleave offers a possibility of very low bit rate compression with high quality of reconstructed audio.

7. REFERENCES

[1] B. Edler, H. Purnhagen, "Parametric audio coding", *Proc. International Conference on Communication Technology*, *ICSP'2000*, vol. 1, pp. 614-617, Beijing, 2000

[2] European Broadcast Union, "EBU subjective listening tests on low-bitrate audio codecs", *EBU Technical Rev*. 3296, June 2003

[3] H. Purnhagen, J. Engdegård, W. Oomen, E. Schuijers, "M10385 Combining low complexity parametric stereo with high efficiency AAC", ISO/IEC JTC1/SC29/WG11 MPEG, Dec. 2003

[4] R. McAulay and T. Quatieri, "Speech Analysis/Synthesis Based on a Sinusoidal Representation", *IEEE Trans. ASSP*, vol. 34, no. 4, pp. 744-754, Aug. 1986

[5] J.S. Marques, L.B. Almeida, "Frequency-varying sinusoidal modeling of speech", *IEEE Trans. ASSP*, vol. 37, no. 5, pp.763-765, 1989

[6] M. Lagrange, S. Marchand, J-B. Rault, "Sinusoidal parameter extraction and component selection in a non-stationary model", *Proc. Int. Conf. on Digital Audio Effects, DAFx'02,* Hamburg, 2002, pp. 59-64

[7] X. Serra, "Musical sound modelling with sinusoids plus noise", in C. Roads et al (eds) *Musical Signal Processing*, Sweets & Zeitlinger, 1997, pp. 91–122

[8] M. Goodwin, "Residual modeling in music analysis/synthesis", *Proc. Int. Conf. Acoustics, Speech and Signal Proc, ICASSP'96*, vol. 2, pp. 1005-1008, May 1996

[9] W. Oomen, A. den Brinker, "Sinusoids plus noise modelling for audio signals", *AES 17th International Conference on High-Quality Audio Coding*, Sep. 1999

[10] A.C. den Brinker, A.W.J. Oomen, "Fast ARMA modelling of power spectral density functions", *Proc. European Signal Proc. Conference EUSIPCO2000*, Tampere, Sept. 2000

[11] T.S Verma, S.N. Levine, T.H-Y Meng, "Transient modelling synthesis: a flexible analysis/synthesis tool for transient signals", *Proc. International Computer Music Conference ICMC'97*, Greece, 1997

[12] R. Badeau, R. Boyer, B. David, "EDS parametric modelling and tracking of audio signal", *Proc. Int. Conf. on Digital Audio Effects, DAFx02*, Hamburg, Sept. 2002

[13] A.C. den Brinker, E.G.P. Schuijers, A.W.J. Oomen, "Parametric Coding for High-Quality Audio", *112th Conv. of the Audio Engineering Society*, Munich, May 2002

[14] ISO/IEC JTC1/SC29/WG11 MPEG, "Int. Standard ISO/IEC 14496-3:2001/AMD2, Sinusoidal Coding", 2004

[15] S. Hainsworth, M. Macleod, "On sinusoidal parameter estimation", *Proc. Int. Conf. on Digital Audio Effects, DAFx03*, London, Sept. 2003

[16] F. Keiler, S. Marchand, "Survey on extraction of sinusoids in stationary sounds", *Proc. Int. Conf. on Digital Audio Effects, DAFx02*, Hamburg, Sept. 2002

[17] M. Abe, J.O. Smith III, "AM/FM rate estimation for time-varying sinusoidal modeling", *Proc. Int. Conf. Acoustics, Speech and Signal Proc, ICASSP'05*, vol. 3, pp. 201-204, 2005

[18] T. Virtanen, "Accurate sinusoidal model analysis and parameter reduction by fusion of components", *Proc. 110th Conv. AES*, Amsterdam, 2001

[19] W. Xue, M. Sandler, "Error compensation in modeling time-varying sinusoids", *Proc. Int. Conf. on Digital Audio Effects, DAFx06*, Montreal, Sept. 2006

[20] G. Meurisse, P. Hanna, S. Marchand, "A new analysis method for sinusoids+noise spectral models", *Proc. Int. Conf. on Digital Audio Effects, DAFx06*, Montreal, Sept. 2006

[21] R. Heusdens, J. Jensen, "Jointly Optimal Time Segmentation, Component Selection and Quantization for Sinusoidal Coding of Audio and Speech", *Proc.ICASSP 2005*, Philadelphia, March, 2005

[22] K. Fitz, L. Haken, „Bandwidth enhanced sinusoidal modeling in Lemur", *Proc. ICMC'95*, Banff, 1995

[23] H.S. Malvar, "A modulated complex lapped transform and its applications to audio processing", *Proc. ICASSP'99*, Phoenix, 1999

[24] J. Princen, A.W. Johnson, A.B. Bradley, "Subband/transform coding using filter bank designs based on time domain aliasing cancellation", *Proc. IEEE Int. Conf. ASSP,* Dallas, Apr. 1987

[25] J.C. Brown, "Musical fundamental tracking using pattern recognition method", *J. Acoust. Soc. Am.,* vol. 92, no. 3, Sept. 1992, pp. 1394-1402

WARPED LINEAR PREDICTION FOR IMPROVED PERCEPTUAL QUALITY IN THE SCELP LOW DELAY AUDIO CODEC (W-SCELP)

Hauke Krüger and Peter Vary

Institute of Communication Systems and Data Processing
RWTH Aachen University, D-52056 Aachen, Germany
krueger@ind.rwth-aachen.de, vary@ind.rwth-aachen.de

ABSTRACT

The SCELP (Spherical Code Excited Linear Prediction) audio codec, which has recently been proposed for low delay audio coding [5], is based on linear prediction (LP). It applies closed-loop vector quantization employing a spherical code which is based on the Apple Peeling code construction rule. Frequency warped signal processing is known to be beneficial especially in the context of wideband audio coding based on warped linear prediction (WLP).
In this contribution, WLP is incorporated into the SCELP low delay audio codec. The overall audio quality of the resulting W-SCELP codec benefits from an improved perceptual masking of the quantization noise. Compared with existing standardized audio codecs with an algorithmic delay below 10 ms, the W-SCELP codec at a data rate of 48 kbit/sec outperforms the ITU-T G.722 codec at a data rate of 56 kbit/sec in terms of the achievable audio quality.

1. INTRODUCTION

Most of the popular audio codecs, e.g. the Advanced Audio Codec (AAC), [1], are based on perceptual audio coding. In perceptual audio coding in general an audio signal is at first transformed by an analysis filter bank. The resulting representation in the transform domain is quantized whereas a perceptual model controls the adaptive bit allocation. Large transform lengths cause a high algorithmic delay. Considering mobile communications, the approach of linear predictive coding (LPC) has been followed for many years in speech coding. In LPC, an all-pole filter models the spectral envelope of an input signal. The signal is filtered with the inverse of that all-pole filter to produce the LP residual which is quantized. In the most recently standardized speech codecs, vector quantization (VQ) based on a sparse codebook is applied, following the CELP (Code Excited Linear Prediction) analysis-by-synthesis principle, [2]. A well-known example for this approach is the adaptive multi rate speech codec (AMR), [3]. Due to the sparseness of the codebook and modeling of the speakers instantaneous pitch period, speech coders can not compete with perceptual audio coding for non-speech input signals. The algorithmic delay is in general lower than that in perceptual coding.
The new SCELP audio codec targets application scenarios which require high audio quality and a very low algorithmic delay, for example digital audio transmission for a wireless headphone. It employs the principle of combined linear prediction and vector quantization (LP VQ) as known from speech coding. In order to achieve a better perceptual audio quality than speech coders, a spherical codebook is employed. The spherical codebook is constructed according to the Apple Peeling principle. This principle was introduced in [4] for the purpose of channel coding. In [5] we have proposed an efficient vector search procedure for the spherical codebook for linear predictive quantization, and in [6] a representation of the available Apple Peeling code vectors as coding trees has been introduced. Both techniques enable very efficient encoding and decoding with respect to computational complexity and memory consumption.
In [7] it was shown that warped signal processing techniques are suited to decrease the required data rate for wideband audio coding while retaining the same subjective audio quality. WLP is employed in a simulated coding system with D*PCM in that contribution. In contrast to that, in this contribution WLP will be incorporated into the closed-loop analysis-by-synthesis framework of the SCELP codec which was introduced for conventional LP primarily.
The principle of the SCELP audio codec and warped linear prediction will be introduced in Section 2 and 3 respectively. The modifications required for the application of WLP in analysis-by-synthesis VQ in general and the SCELP framework for highly efficient encoding in particular are described in Section 4. Results are presented in Section 5, including a comparison of the W-SCELP codec with the ITU-T G.722 [8] low delay audio codec.

2. PRINCIPLE OF THE SCELP AUDIO CODEC

The SCELP low delay audio codec is based on block adaptive combined linear prediction and vector quantization: The correlation immanent to an input signal $x(k)$ is exploited

in order to achieve a high quantization signal-to-noise-ratio (SNR). For this purpose, a windowed segment of the input signal of length L_{LP} is analyzed in order to obtain the N time-variant filter coefficients $a_1 \cdots a_N$. Based on these LP coefficients the LP analysis filter with system function $H_A(z) = 1 + \sum_{i=1}^{N} a_i \cdot z^{-i}$ converts the input signal into the LP residual signal $d(k)$ which is segmented into $N_V = L_{LP}/L_V \in \mathbb{N}$ non overlapping signal vectors $\mathbf{d} = \begin{bmatrix} d_0 & d_1 & \cdots & d_{L_V-1} \end{bmatrix}$ of length L_V. Each LP residual vector is quantized and transmitted to the decoder as code vector index i_Q. For signal reconstruction, also the LP coefficients must be transmitted to the decoder. In general this can be realized with only small additional bit rate as shown for example in [9]. In the decoder, the transmitted code vector index i_Q is the basis for the reconstruction of the quantized LP residual vector $\tilde{\mathbf{d}}$ which is filtered by the LP synthesis filter $H_S(z) = (H_A(z))^{-1}$. The output of the LP synthesis filter is the decoded signal vector $\tilde{\mathbf{x}}$ and hence the signal $\tilde{x}(k)$ [10].
The principle of the encoder of a CELP codec is depicted in Figure 1. The decoder is part of the encoder. According to

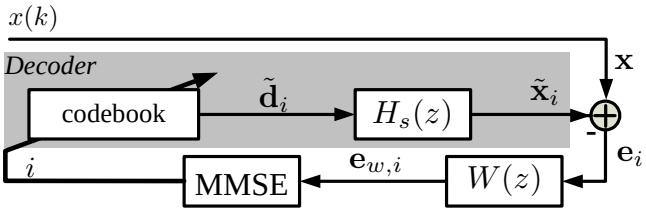


Figure 1: Scheme SCELP Audio Codec, Encoder.

the analysis-by-synthesis principle, the LP residual vector $\tilde{\mathbf{d}}_i$ for each codebook index i is generated first. This excitation vector is filtered by the LP synthesis filter $H_S(z)$ to obtain the corresponding decoded signal vector candidates $\tilde{\mathbf{x}}_i$. The error distance between the input signal and the decoded signal, $\mathbf{e}_i = \mathbf{x} - \tilde{\mathbf{x}}_i$, is determined for each vector candidate corresponding to index i. The goal is to find the index i_Q for which the minimum mean square error is achieved:

$$i_Q = \arg\min_i \{\mathcal{D}_i = \| \mathbf{e}_i \|^2 = (\mathbf{x} - \tilde{\mathbf{x}}_i) \cdot (\mathbf{x} - \tilde{\mathbf{x}}_i)^T\}. \quad (1)$$

The error weighting filter $W(z)$ controls the spectral shape of the quantization noise inherent to the decoded signal for perceptual masking of the quantization noise. The analysis-by-synthesis vector search can be exhaustive for a large vector codebook.

2.1. Spherical Vector Codebook

In the SCELP audio codec, vector quantization is applied in a *gain-shape* approach to encode the LP residual. Each LP residual vector $\mathbf{d}$ is decomposed into a radius for the *gain* and a vector on the surface of a unit sphere for the *shape* component. While the radius R is quantized by means of logarithmic scalar quantization, the valid code vectors for the quantization of the shape component are based on the Apple Peeling code construction rule. This rule was described and demonstrated for the special case of a 3-dimensional sphere in [5]. The design target of the Apple Peeling code is to place all codebook vectors on the surface of a unit sphere as uniformly as possible.
The **decoder** in CELP coding in general is not very complex. For a low computational **encoding** complexity, the analysis-by-synthesis approach in Figure 1 was modified in the SCELP encoder as described in [5]. The result is a low complexity vector search framework. Additionally, the technologies called *Pre-Selection* and *Candidate-Exclusion*, combined with an efficient metric computation, enable a very efficient code vector search. Furthermore the representation of the Apple Peeling code vectors as coding trees was explained in [6] for the sake of highly efficient encoding and decoding.

3. WARPED LINEAR PREDICTION

The principle and properties of warped linear prediction are discussed in [7]. In this contribution only those aspects that are relevant for the analysis-by-synthesis vector search of the SCELP will be briefly presented.
In conventional linear prediction the approximation of the spectral envelope of a signal is based on a uniform resolution of the frequency scale. Considering the perceptual properties of human hearing, a uniform resolution is known to be inferior compared to a non-uniform resolution of the frequency scale. For this purpose, a non-uniform resolution of the frequency scale is achieved by applying WLP. Considering the z-transform of a signal, this can be realized by replacing all unit delay elements by an allpass filter $AP(z)$,

$$z^{-1} \rightarrow AP(z) = \frac{z^{-1} - \lambda}{1 - \lambda \cdot z^{-1}} \quad | \lambda | < 1; \lambda \in \mathbb{R} \quad (2)$$

For positive values of warping constant λ, the spectral resolution is increased for lower and decreased for higher frequencies compared to conventional LP.

3.1. Warped LP Analysis

In the SCELP codec the LP analysis is based on the auto correlation method, as for example described in [10]. In [7], it was shown that for the warped LP analysis, in the auto correlation method all unit delay elements must be replaced by the first order allpass filter $AP(z)$ according to (2). Hence the warped auto correlation coefficients $\varphi^w_{x,x}(0) \cdots \varphi^w_{x,x}(N)$ are determined as demonstrated for the first three coefficients in Figure 2. Warped auto correlation coefficients can

be transformed into warped LP coefficients $a_1^w \cdots a_N^w$ by means of the Levinson Durbin algorithm as in conventional LP.

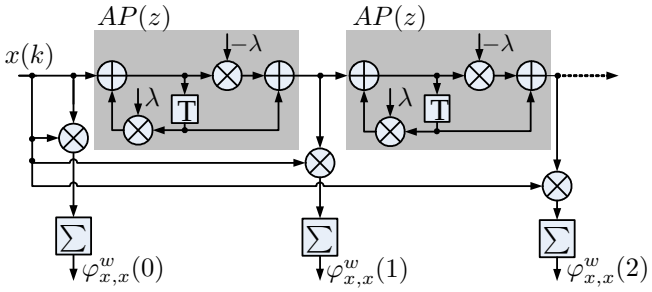


Figure 2: Warped LP Analysis.

3.2. LP Analysis/Synthesis filter

For the warped LP analysis and synthesis filter, all unit delay elements of the conventional LP analysis/synthesis filter are replaced by allpass filters $AP(z)$:

$$H_A^w(z) = H_A^w(AP(z)) = 1 + \sum_{i=1}^{N} a_i^w \cdot AP(z)^i = (H_S^w(z))^{-1}. \tag{3}$$

The filter coefficients a_i^w are calculated according to Section 3.1.

3.3. Error Weighting Filter

The SCELP audio codec employs an error weighting filter as proposed in [11]. In conventional linear prediction this error weighting filter can be calculated from the LP analysis filter:

$$W(z) = \frac{H_A(z/\gamma_2)}{H_A(z/\gamma_1)}. \tag{4}$$

The coefficients γ_1 and γ_2 are within the range of $0 \leq \gamma_1 \leq \gamma_2 \leq 1.0$ and control the degree of noise shaping. With the application of the error weighting filter, the quantization noise inherent to the decoded output signal is spectrally shaped according to the system function of the inverse of the error weighting filter, $(W(z))^{-1}$.
Considering WLP, all unit delay elements in equation (4) must be replaced by $AP(z)$ in the warped error weighting filter:

$$W^w(z) = \frac{H_A^w(AP(z) \cdot \gamma_2)}{H_A^w(AP(z) \cdot \gamma_1)}. \tag{5}$$

4. WLP IN THE SCELP CODEC

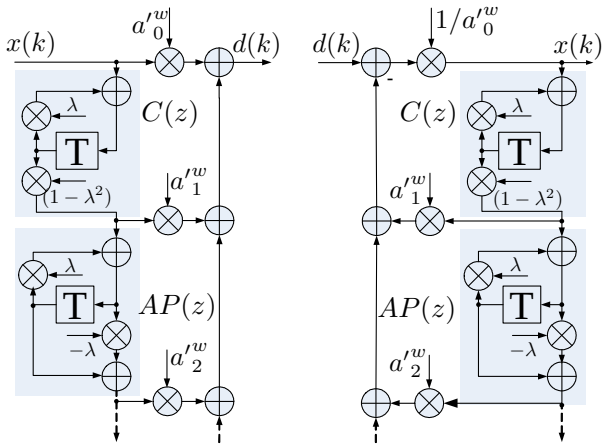
The properties of the warped linear prediction prohibit a straight forward incorporation into the SCELP audio codec. Therefore the following modifications must be considered first.

4.1. Zero-Delay Path in Feedback Loop

A zero-delay path in the feedback loop makes the implementation of the LP synthesis filter according to equation (3) impractical. In contrast to the implementation of the warped LP synthesis filter in [7], in this contribution [12] the substitution of

$$C(z) = AP(z) + \lambda \tag{6}$$

is applied to the first allpass filter in the allpass chain of the warped LP synthesis filter to remove the zero-delay path. The resulting filter structure is employed for warped LP analysis and synthesis filter as depicted in Figure 3, and also for the error weighting filter (5).



(a) WLP Analysis Filter (b) WLP Synthesis Filter

Figure 3: Modified Structure for Warped LP Filters.

As a consequence of the applied substitution, modified filter coefficients $a'^w_0 \cdots a'^w_N$ are used in the new LP analysis and synthesis filter structure. These can be calculated from the original coefficients $a_0^w \cdots a_N^w$ recursively as

$$\begin{aligned} a'^w_N &= a_N^w \\ a'^w_i &= a_i^w - \lambda \cdot a'^w_{i+1}; \quad i = N-1, \cdots 0; \; a_0^w = 1. \end{aligned} \tag{7}$$

4.2. Zero-Mean Property

Decorrelation of an input signal without any additional amplification is connected to the well-known zero-mean property in conventional LP [13]. WLP does not provide this

property. It can be shown, however, that the WLP filters have zero-mean property if the modified filter coefficients a'^{w}_{i} (7) are normalized according to

$$a''^{w}_{i} = a'^{w}_{i} / a'^{w}_{0} \qquad i = N, \cdots, 0. \tag{8}$$

Considering Figure 3, the filter coefficients a'^{w}_{i} must be replaced by the normalized coefficients a''^{w}_{i}, with the first coefficient resulting to $a''^{w}_{0} = 1.0$.

4.3. Spectral Tilt Compensation

Due to the non uniform resolution of the frequency scale in warped signal processing, the warped LP residual signal is not perfectly flat but contains a spectral tilt [14]. This spectral tilt inherent to the LP residual must be compensated prior to quantization to achieve the highest quantization SNR in closed-loop LP VQ. For this purpose the filters according to (3), in the structure as depicted in Figure 3, employing the normalized coefficients a''^{w}_{i} (8), are operated in the cascade with the tilt compensation filter

$$H^{w}_{t}(z) = 1 - \lambda \cdot z^{-1} \tag{9}$$

to form the overall LP analysis/synthesis filter [1]

$$H^{w,t}_{A}(z) = H^{w}_{A}(z) \cdot H^{w}_{t}(z) = (H^{w,t}_{S}(z))^{-1}. \tag{10}$$

4.4. SCELP Low Complexity Vector Search

Considering the analysis-by-synthesis principle, a low complexity vector search procedure is employed in the SCELP codec. It enables to search the large spherical vector codebook in a very efficient way to achieve the low computational complexity of the codec. The principle was introduced in [5] and is depicted in Figure 4.

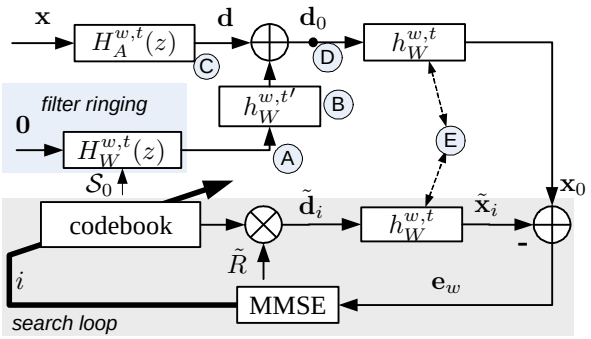


Figure 4: Modified Analysis-by-synthesis.

[1] No tilt compensation is required for the error weighting filter because the same warping factor is used for numerator and denominator part.

Considering Figure 4, those functional blocks which must be adapted for the application of WLP will be identified in the following:
The basis for an efficient vector search in the SCELP is the determination of signal $\mathbf{d}_0$ prior to the actual analysis-by-synthesis vector search procedure. In order to determine this signal, the filter ringing signal, marked by the label A, must be obtained first. Latter is the filter output resulting from the history related to previously quantized signal frames. Before a new signal input vector $\mathbf{x}$ is quantized, this history is stored as the filter states $\mathcal{S}_0$. In order to get the vector related to the filter ringing, the filter $H^{w,t}_{W}(z)$ is fed with a zero input signal vector $\mathbf{0}$ of length L_V. For the W-SCELP, this filter is identified as the cascade of the warped LP synthesis filter according to (10) and the error weighting filter according to (5):

$$H^{w,t}_{W}(z) = H^{w,t}_{S}(z) \cdot W^{w}(z). \tag{11}$$

In order to obtain signal vector $\mathbf{d}_0$, the filter ringing signal vector must be transformed into the residual signal domain. This is done in Figure 4 by means of convolution, marked as the block $h^{w,t'}_{W}$ at position B. $h^{w,t'}_{W}$ is identified as the truncated impulse response of the inverse of filter $H^{w,t}_{W}(z)$ (11):

$$h^{w,t'}_{W} = \left[h^{w,t\,\prime}_{W,0} \cdots h^{w,t\,\prime}_{W,(L_V-1)} \right]; \quad h^{w,t\,\prime}_{W,k} \circ\!\!-\!\!\bullet\, (H^{w,t}_{W}(z))^{-1} \tag{12}$$

The convolved filter ringing is added to the warped LP residual $\mathbf{d}$. The LP residual $\mathbf{d}$ is obtained by filtering $\mathbf{x}$ in the warped LP analysis filter $H^{w,t}_{A}(z)$, position C. The resulting signal vector $\mathbf{d}_0$, position D, is analyzed to determine the corresponding radius R which is quantized as $\tilde{R}$.
In the analysis-by-synthesis vector search procedure, code vectors $\tilde{\mathbf{d}}_i$ are generated by multiplying the spherical shape-component vector candidates with the quantized radius $\tilde{R}$. Considering the metric (1) to find the optimal excitation vector $\tilde{\mathbf{d}}_{i_Q}$, the unquantized and the quantized residual vector candidate, $\mathbf{d}$ and $\tilde{\mathbf{d}}_i$, both must be transformed from the LP residual into the signal domain. For this purpose, the two blocks $h^{w,t}_{W}$, position E, represent the transform for both signals by means of convolution with the truncated impulse response of filter $H^{w,t}_{W}(z)$ (11):

$$h^{w,t}_{W} = \left[h^{w,t}_{W,0} \cdots h^{w,t}_{W,(L_V-1)} \right]; \quad h^{w,t}_{W,k} \circ\!\!-\!\!\bullet\, H^{w,t}_{W}(z) \tag{13}$$

Now that all functional blocks, which were introduced for the SCELP codec, have been identified also for the W-SCELP codec, the principles of *Pre-Selection*, the efficient metric computation and the *Candidate-Exclusion* explained in [5] for highly efficient encoding can be applied also in the W-SCELP.
With the determination of quantized LP residual vector $\tilde{\mathbf{d}}_{i_Q}$,

the differential signal $\mathbf{d} - \tilde{\mathbf{d}}_{i_Q}$ must be processed by filter $H_W^w(z)$ to finally determine the update for the filter states $\mathcal{S}_0$ restored for the quantization of the next signal frame.

4.5. Complexity

The computational complexity of the warped LP analysis, synthesis and error weighting filter in the W-SCELP codec is higher than that of the same filters realized for conventional LP in the SCELP codec. The biggest part of the overall complexity of the SCELP codec, however, is spent on the analysis-by-synthesis vector search. Since the W-SCELP benefits from the same principles targeting low complexity encoding as the SCELP, the overall complexity is only marginally increased. The complexity of the encoder of the conventional SCELP codec was estimated as 20-25 WMOPS in [5], that of the encoder of the W-SCELP codec as 23-28 WMOPS. The decoder of the W-SCELP codec has an estimated complexity of 2-3 WMOPS.

5. RESULTS

For the comparison of the achieved quality of the W-SCELP and the SCELP codec, both codecs have been configured identically for a sample rate of $f_s = 16$ kHz. The resulting overall data rate is approximately 48 kbit/sec, and the noise shaping coefficients have been set to $\gamma_1 = 0.6$ and $\gamma_2 = 0.94$. The order of the linear prediction in both cases is $N = 10$ and the algorithmic delay $L_{LP}\hat{=}9$ ms. For the W-SCELP codec, the highest performance has been determined for a warping factor $\lambda = 0.46$ in informal listening tests.

Comparing the prediction gain in W-SCELP and SCELP as a measure of signal decorrelation, WLP provides only an insignificantly higher value. Considering perceptual masking of the quantization noise, it was observed in informal listening tests that the higher spectral resolution of WLP for lower frequencies provides significant benefits. Especially for audio signals with a sparse spectrum, for example the sound of a flute, WLP provides clearly better perceptual results than conventional LP.

Considering a formal assessment of the quality, speech was processed by the W-SCELP and the SCELP codec. The decoder output was rated with the WB-PESQ measure [15] which is widely used in the speech coding community. As result, the W-SCELP outperformed the SCELP by 0.2 on the MOS scale. Comparable results may also be obtained using the PEAQ quality measure [16].

For a comparison of the W-SCELP codec with a standardized audio codec, the same speech signal was also processed by the ITU-T G.722 low delay audio codec at 48, 56 and 64 kbit/sec. This reference codec was chosen because of its algorithmic delay in the magnitude of that of the W-SCELP codec (below 10 ms)[2]. The result of the formal comparison of the new codec with the G.722 reference codec is listed in Table 1 in the order of descending perceptual quality. The

Codec	G.722 mode 1	**W-SCELP**	G.722 mode 2	G.722 mode 3
Data rate	$64\frac{\text{kBit}}{\text{sec}}$	$48\frac{\text{kBit}}{\text{sec}}$	$56\frac{\text{kBit}}{\text{sec}}$	$48\frac{\text{kBit}}{\text{sec}}$
WB-PESQ (MOS-LQO)	4.47	4.4	4.39	4.02

Table 1: Results Formal Quality Assessment.

performance of the G.722 codecs was rated with 4.02, 4.39 and 4.47 MOS for the three codec modes respectively. The W-SCELP at a data rate of roughly 48 kbit/sec reached a value of 4.4 MOS. Considering this result, the quality of the W-SCELP codec at 48 kbit/sec can be classified as slightly better than that of the G.722 at 56 kbit/sec.

6. CONCLUSION

In this contribution the principle of warped signal processing was incorporated into the new SCELP low delay audio codec to form the W-SCELP codec. While the overall complexity of the W-SCELP is only insignificantly higher than that of the SCELP codec, the achievable audio quality is clearly better. In a comparison with a standardized codec that has a similar algorithmic delay, the W-SCELP at a data rate of 48 kbit/sec outperforms the ITU-T G.722 audio codec at a data rate of 56 kbit/sec.

7. REFERENCES

[1] ISO/IEC 13818-7, "Advanced Audio Coding (AAC)," 1997.

[2] M. Schroeder and B. Atal, "Code-excited Linear Prediction (CELP): High-quality Speech at very low Bit Rates," *Proc. ICASSP*, 1985.

[3] Rec. GSM 06.90 ETSI, "Adaptive Multi-Rate (AMR) Speech Transcoding," 1998.

[4] E. Gamal, L. Hemachandra, I. Spherling, and V. Wei, "Using Simulated Annealing to Design Good Codes," *IEEE Trans. Inform. Theory*, vol. it-33, 1987.

[5] H. Krüger and P. Vary, "SCELP: Low Delay Audio Coding with Noise Shaping based on Spherical Vector Quantization," *EUSIPCO, Florence, Italy*, 2006.

[2] An alternative codec with a comparable algorithmic delay is the ULD codec [17]. This codec, however, is not freely available and was thus not considered.

[6] H. Krüger and P. Vary, "An Efficient Codebook for the SCELP Low Delay Audio Codec," *MMSP, Victoria, Canada*, 2006.

[7] A. Härmä and U. Laine, "A Comparison of Warped and Conventional Linear Predictive Coding," *IEEE Trans. Speech and Audio Processing*, vol. 9, no.5, 2001.

[8] ITU-T Rec. G.722, "7 kHz Audio Coding within 64 kbit/s," 1988.

[9] K. Paliwal and B. Atal, "Efficient Vector Quantization of LPC Parameters at 24 Bits/Frame," *IEEE Trans. Speech and Signal Proc.*, vol. 1, no.1, pp. 3–13, 1993.

[10] N. Jayant and P. Noll, *Digital Coding of Waveforms*, Prentice-Hall, Inc., 1984.

[11] M. Schroeder, B. Atal, and J. Hall, "Optimizing Digital Speech Coders by Exploiting Masking Properties of the Human Ear," *Journal of the Acoustical Society of America*, pp. 1647–1652, 1979.

[12] K. Steiglitz, "A Note on Variable Recursive Digital Filters," *IEEE Trans. Acc., Speech, and Signal Processing*, vol. 28, 1980.

[13] J. Markel and A.Gray, *Linear Prediction of Speech*, Springer, 1976.

[14] H. W. Strube, "Linear Prediction on a Warped Frequency Scale," *Journal of the Acoustical Society of America*, vol. 68, pp. 1071–1076, 1980.

[15] ITU-T Rec. P.862.2, "Wideband Extension to Recommendation P.862 for the Assessment of Wideband Telephone Networks and Speech Codecs," 2005.

[16] Recommendation ITU-R BS.1387, "Method for objective measurements of perceived audio quality," 2001.

[17] http://www.idmt.fraunhofer.de, "Audio Coding with Ultra Low Encoding/Decoding Delay," 2007.

TRANSIENT ENCODING OF AUDIO SIGNALS USING DYADIC APPROXIMATIONS

Francois Xavier Nsabimana and Udo Zölzer

Helmut-Schmidt-University / University of the Federal Armed Forces
Department of Signal Processing and Communications
Hamburg, Germany
fransa, udo.zoelzer@hsu-hh.de

ABSTRACT

In this paper, we present a frame based approach for transient detection and encoding of audio signals. The transient detection procedure, as presented here, uses linear prediction within a signal frame followed by an envelope estimation to build an adaptive threshold. Detected transients will automatically be separated and the gaps left by the removed transient are filled with samples from forward and backward extrapolation. To encode detected transients, dyadic approximation approaches are discussed. Results of the application to different audio signals are also presented.

1. INTRODUCTION

An audio signal is generally composed of two parts: a deterministic and a stochastic part. The deterministic part of an audio signal consists of sinusoids, while noise and transients constitute the stochastic part. Although some models as proposed in [1, 2, 3] represent well sinusoids and noise, they really fail for transients. Since transients do not fit well into sinusoids and noise models, they therefore need their own model. In order to build a three components approach (transients + sinusoids + noise) of an audio signal, we need to split the stochastic part into two parts.
In [1] a model for sinusoids with time-varying amplitudes, phases and harmonic frequencies has been presented. In [2], the sinusoidal model (SM) proposed in [1] was extended with a noise model based on residual approximation. The Spectral Modelling Synthesis (SMS) presented in [2] gives good results when applied to audio signals only composed of sinusoids and noise. But once transients occur in an audio signal, they will then appear in the residual signal. This will thus raise the spectral envelope of the noise during a residual approximation, yielding a synthesized signal with artefacts. To avoid this, a pre-processing step is required to first separate the transient's contribution.
In [4, 5, 6, 7, 8] and many other recent publications, transient representation is investigated. The methods proposed can be classified into three categories: time domain approach, frequency domain approach and hybrid approach [9]. In this paper, we address the problem of a transient representation under the hybrid approach. We first perform detection in time domain based upon linear prediction followed by encoding in frequency domain using dyadic approximation. In our three components approach (Fig. 1), sinusoids and noise are represented using techniques proposed in [2, 10].
In Section 2, we will present our transient detection approach, Section 3 will introduce the transient encoding approaches, Section 4 will show some simulation results, we will finally close with a conclusion and outlook in Section 5.

2. TRANSIENT DETECTION

Since transients are poorly represented by sinusoids or noise model, it is preferable to represent them separately and leave sinusoids and noise to their own models. We propose, in Fig. 1, to first detect and separate transients within a signal frame. Transients represented as filtered noise will certainly lose their sharpness and sound bad. In Fig. 2 the pre-processing steps for the transient detection are detailed. Transients can be classified into two categories: visible and hidden transients. The main difficulty does not consist of detecting visible transients, but those of small energy. A good transient detector should thus reveal the presence of hidden transients and then emphasize them. In order to successfully detect both kinds of transients, we need to apply a filter which should absorb most of the audio signals (sinusoids and noise) energy leaving transients unchanged. Since sudden changes in audio signals, like transients, remain unpredictable, the prediction error will then accentuate transients in the audio signal.

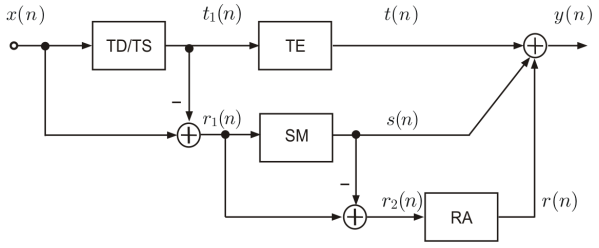


Figure 1: *Transients + Sinusoids + Noise approach. TD/TS (Transient Detection / Transient Separation), TE (Transient Encoding), SM (Sinusoidal Modelling), RA (Residual approximation). s(n): sinusoids, t(n): transients, r(n): noise.*

2.1. Linear Prediction

Assume that signal sample $x(n)$ of an audio signal is to be estimated combining p previous samples. The estimated sample $\hat{x}(n)$ will then be obtained using a finite impulse response (FIR) filter given by

$$\hat{x}(n) = \sum_{i=1}^{p} a_i \cdot x(n-i), \tag{1}$$

where a_i are the filter coefficients obtained by minimizing the square of the prediction error

$$e(n) = x(n) - \hat{x}(n) = x(n) - \sum_{i=1}^{p} a_i \cdot x(n-i) \tag{2}$$

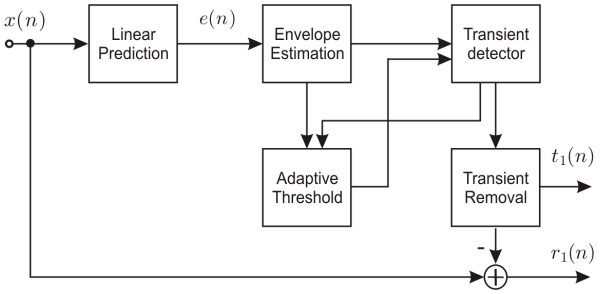


Figure 2: *TD/TS: Transient Detection & Separation approach.* $t_1(n)$*: detected transients,* $r_1(n)$*: first residual.*

within a signal frame. Transforming Eq. (2) into the Z-domain, we obtain the following filter transfer function

$$A(z) = \frac{E(z)}{X(z)} = 1 - P(z) = 1 - \sum_{i=1}^{p} a_i \cdot z^{-i}. \qquad (3)$$

The filter $A(z)$ is designed so that all its zeros are inside the unit circle $|z| = 1$ (excluding the unit circle itself). If we need to recover the filtered input signal $x(n)$ from the error signal $e(n)$, we then have to apply the inverse (synthesis) filter

$$H(z) = \frac{Y(z)}{E(z)} = \frac{1}{1 - P(z)} = \frac{1}{A(z)}. \qquad (4)$$

on $e(n)$.

2.2. Envelope estimation

The simple way to estimate the temporal envelope of a signal $x(n)$ is to take the absolute value of $x(n)$ and apply smoothing using low-pass filter or peak detector. In our model (see Fig. 2), we choose a common and very efficient technique based on the Hilbert Transform (a 90 degree phase shifter) to estimate the envelope of the prediction error signal. With that technique the envelope of a signal $x(n)$ is computed using the corresponding analytic signal

$$\tilde{x}(n) = x(n) + j \cdot \hat{x}(n), \qquad (5)$$

where $\hat{x}(n)$ is the Hilbert transform of $x(n)$. The envelope of the original signal is then simply the modulus of the analytic signal given by

$$x_{env}(n) = |\tilde{x}(n)| = \sqrt{x^2(n) + \hat{x}^2(n)}. \qquad (6)$$

Finally, the envelope is smoothed using a first order low-pass filter.

2.3. The proposed method

The main steps for transient detection as depicted in Fig. 2 can be described as follows. An input signal $x(n)$ is decomposed into short frames of 1024 samples with an overlap of 512 samples. Linear prediction is then applied in each frame to reveal the transients locations. A prediction filter with model order $p = 8$ is sufficient for our application. A suitable envelope estimator is applied to the prediction error to build the threshold. The threshold function will be kept constant in transient area and follows the error signal elsewhere. This is done by comparing the actual value of the envelope with the weighted mean value of the envelope from the previous frame. The weighting factor is obtained by dividing the maximum value with the mean value of the envelope in the current frame. If the value of the envelope is higher than the weighted mean value, the threshold function is kept constant equal to the non-weighted mean value of the envelope from the previous frame. A binary sequence is then set to one at those indexes where a transient is occurring in the current frame (see Fig. 3 - 4 lower right). In order to avoid multiple detections of the same transient in the overlapping zone, a strategy has been developed. That is if a transient is fully embedded in a frame, a detection flag is triggered. If a fully embedded transient spans over both half-frame, the index of the detected transient corresponding to the second half-frame are used in the next frame for comparison. If a transient spans over successive frames, a detection flag is not triggered since the transient data over these frames needs to be assembled. The gaps left when detected transients are removed, are filled with samples using forward and backward extrapolation as presented in [11]. A three components approach is built, sinusoids and noise are thus left to their respective models, while transients are encoded separately.

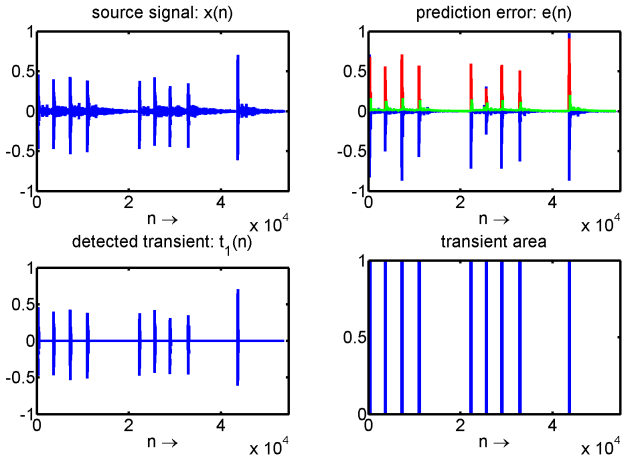


Figure 3: *Transients detection in Castanets sound file.*

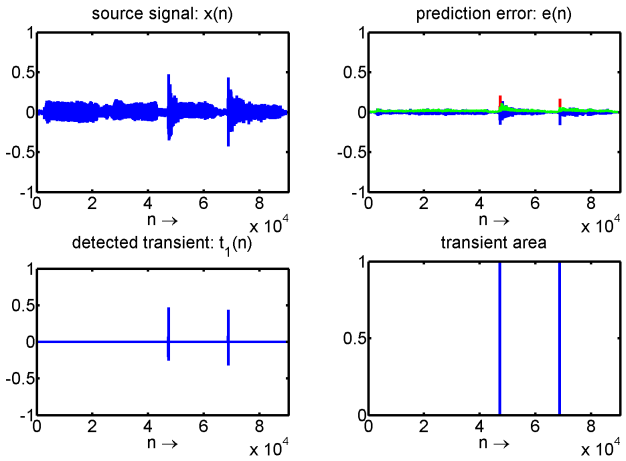


Figure 4: *Transients detection in ABBA sound file.*

2.4. Signal extrapolation

In [11, 12] extrapolation and signal restoration of damaged or removed samples have been deeply investigated. The underlying idea is here the same like in linear prediction, with the only difference that extrapolation needs forward and backward predictor. In our approach the samples to be extrapolated are those corresponding to the removed transients samples (N_2). The procedure for the extrapolation of the missing samples is depicted in Fig. 5, while the results are shown in Fig. 6. The main steps of the extrapolation procedure can be explained as follow:

- Determine the number N_2 of missing samples x_2 within a frame.
- Compare the number N_1 and N_3 of known signal samples (x_1 and x_3) with the number N_2 of missing samples x_2.
- For forward extrapolation: if there are fewer known signal samples than missing samples ($N_1 < N_2$), take known samples from previous frame.
- For backward extrapolation: if there are fewer known signal samples than missing samples ($N_3 < N_2$), take known samples from next frame. In this case backward extrapolation is done in the next frame.
- Apply autoregressive (AR) model of order $p \leq N_2$ to calculate the filter coefficients: a_{if} and a_{ib}.
- Initialize the filter with p past known samples just before the section to be extrapolated.
- Generate a vector of zeros with length N_2, feed it together with z_{if} or z_{ib} as input to the extrapolation filters.
- The output of the two filters are the N_2 extrapolated samples (x_{ef} and x_{eb}).
- Finally sum the forward extrapolated samples and backward extrapolated samples weighted with an appropriate window function (see Fig. 6).

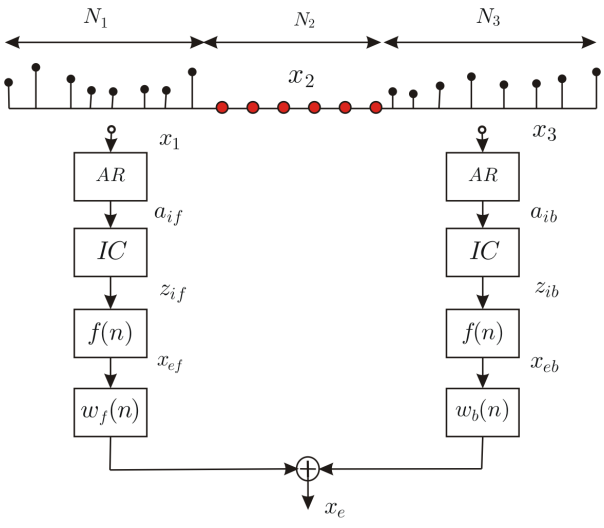


Figure 5: *Extrapolation procedure: autoregressive parameter estimation (AR), Initial Conditions for filter implementation (IC), filter (f(n)), appropriate window function ($w_f(n), w_b(n)$).*

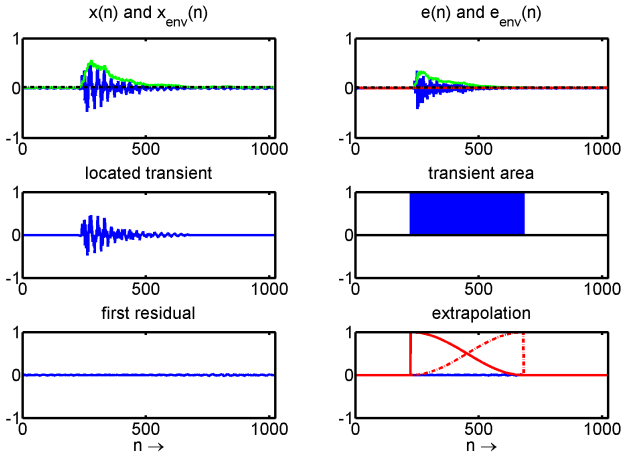


Figure 6: *Transients detection in Castanets sound file.*

3. TRANSIENT ENCODING

We have already stated that transients do not fit well into sinusoids and noise models. Since transients need a very good time resolution, while sinusoids require a very good frequency resolution, a transformation which works with variable resolution is therefore needed. Methods based on dyadic approximation such us multiscale approximation or octave splitting, as presented in Fig. 7, address the problem of multiresolution. An important dyadic approximation method, which has gained increasing attention during the last years, is the discrete wavelet transform (DWT). In [5, 6] transients detection and encoding based on wavelet transform and Hidden Markov tree are presented, while in [13, 14] methods based on iterated filter bank are used. Let $|X(f)|$ be the magnitude spectrum of a signal $x(n)$, using dyadic approximation, the spectrum of $x(n)$ can be first split into two equal parts: low-pass band and high-pass band. These two bands can again be decomposed into

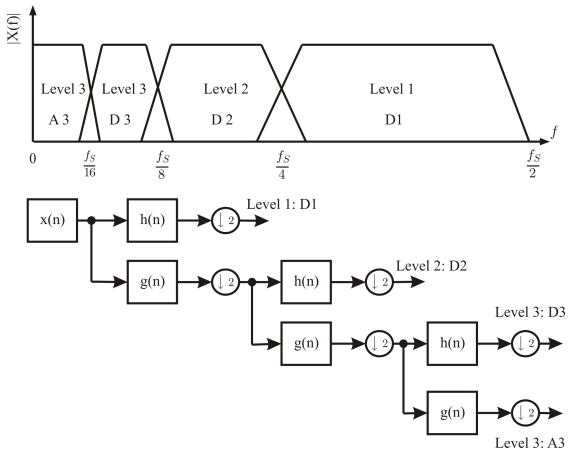


Figure 7: *DWT in frequency domain upper, a 3 level filter bank lower.*

subbands until we reach the number of bands needed for the application (see Fig. 7). Since discrete wavelet transform (DWT) is an implementation of wavelet transforms as an iterated filter bank,

the LP and HP, within one level (see Fig. 7) will represent respectively the scaling filter and the wavelet filter. The signal $x(n)$ is simultaneously decomposed using low-pass filter g and high-pass filter h (see Fig. 7 lower) yielding approximation coefficients

$$A_j(k) = \sum_{i=1}^{N_x+N_F-1} x(i) \cdot g(2k-i) = (x * g)(n) \downarrow 2, \quad (7)$$

and detail coefficients

$$D_j(k) = \sum_{i=1}^{N_x+N_F-1} x(i) \cdot h(2k-i) = (x * h)(n) \downarrow 2, \quad (8)$$

where j is the index of the considered subband, while N_x and N_F are respectively the length of the input signal $x(n)$ and the length of the impulse response g or h (see Fig. 8). The filters g and h are related to each other and must satisfy the quadrature mirror filter relationship.

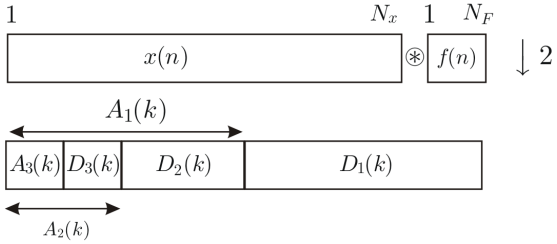


Figure 8: *A 3 level Wavelet decomposition, decomposition filter f(n) (g(n) , h(n)), input signal x(n), detail coefficients ($D_j(k)$) and approximation coefficients ($A_j(k)$).*

In this section, we present our transient encoding approach which performs quantization of the prediction error from discrete cosine transform (DCT) applied on DWT coefficients. We will compare various combinations of dyadic approximations applied to the detected transient. The first method will combine discrete wavelet transform (DWT) and coefficients thresholding within each band. The second method is our proposed approach. The third method is a modified version of the second method, where the discrete cosine transform is not applied. The last method deals with quantization of the prediction error from DCT transformed transient signal. Regarding fame, minimum of smoothness and reduced number of coefficients, we have chosen Daubechies wavelets with 4 vanishing moments (zero moments) to decompose and reconstruct the transients.

3.1. Method I - DWT / Thresholding

This method, as depicted in Fig. 9, deals with discrete wavelet transform (DWT), coefficient thresholding within a subband and inverse discrete wavelet transform (IDWT) for signal reconstruction. The aim of this method is to investigate how far we can reduce the number of coefficients in each subband, but still be able to reconstruct the decomposed signal with the few retained coefficients after thresholding. We have seen from the decomposition explained with Fig. 8, that the approximation coefficients are again split into two parts yielding new detail coefficients and new approximation coefficients of the corresponding subband. Regarding importance of the approximation coefficients in the last subband, we decide not to threshold $A_3(k)$. In the analysis part, a detected transient will first be discrete wavelet transformed using Daubechies wavelets (db4: here 4 is for the order or the vanishing moments) with 3 levels. Applying thresholding (Thresh) based on Root Mean Square (RMS)

$$RMS = \sqrt{\frac{1}{N}\sum_{k=1}^{N} D_j^2(k)}, \quad (9)$$

in each band, except the low-pass band (A_3) (see Fig. 7), we have considerably reduced the number of coefficients to be used for reconstruction. To threshold the wavelets coefficients we have compared three functions for wavelets coefficients thresholding. The hard-thresholding

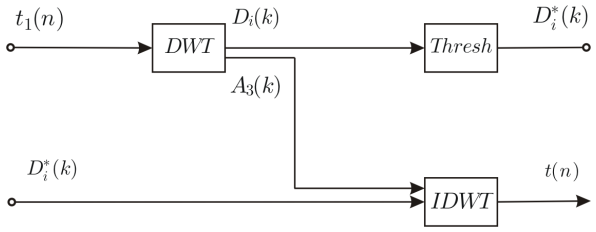


Figure 9: *Method I: Discrete Wavelet Transform (DWT), Thresholding (Thresh) and Inverse Discrete Wavelet Transform (IDWT).*

$$f_H(D_j) = \begin{cases} D_j & \text{if } |D_j| \geq \lambda, \\ 0 & \text{elsewhere.} \end{cases} \quad (10)$$

retains all coefficients that are greater than the chosen threshold value λ (RMS) and sets to zero others. The soft-thresholding

$$f_S(D_j) = \begin{cases} D_j - \lambda & \text{if } D_j \geq \lambda, \\ 0 & \text{if } |D_j| < \lambda \\ D_j + \lambda & \text{if } D_j \leq -\lambda \end{cases} \quad (11)$$

is shrinkage function, since it shrinks the coefficients by λ towards zero. From Fig. 10 we can see that the hard-thresholding (blue curve) is discontinuous at $|D_j| = \lambda$, while soft-thresholding (green curve) is continuous at $|D_j| = \lambda$ but modifies the value of the retained coefficients. In [15] a custom thresholding function

$$f_C(x) = \begin{cases} D_j - sgn(D_j)(1-\alpha)\lambda & \text{if } |D_j| \geq \lambda, \\ 0 & \text{if } |D_j| \leq \gamma \\ \alpha\lambda(\frac{|D_j|-\gamma}{\lambda-\gamma})^2(\alpha-3)(\frac{|D_j|-\gamma}{\lambda-\gamma}) + 4 - \alpha & \text{elsewhere} \end{cases} \quad (12)$$

where $0 < \gamma < \lambda$ and $0 \leq \alpha \leq 1$ is proposed. This function is a linear combination of hard-thresholding and soft-thresholding, since it combines the advantages of both functions. In Fig. 10, we can recognize that custom-thresholding is equivalent to hard-thresholding with smooth transition around the threshold. From Eq. (12) we can easily go back to Eq. (11) by taking $\alpha = 0$ and to Eq. (10) by taking $\alpha = 1$ and $\gamma = \frac{\lambda}{2}$. For this method, we have finally applied the custom-thresholding function.
The decomposition in subbands is shown in Fig. 9. Except the lower band (A3), the thresholding concerns here only the upper bands. The results of the reconstruction are shown in Fig. 16.

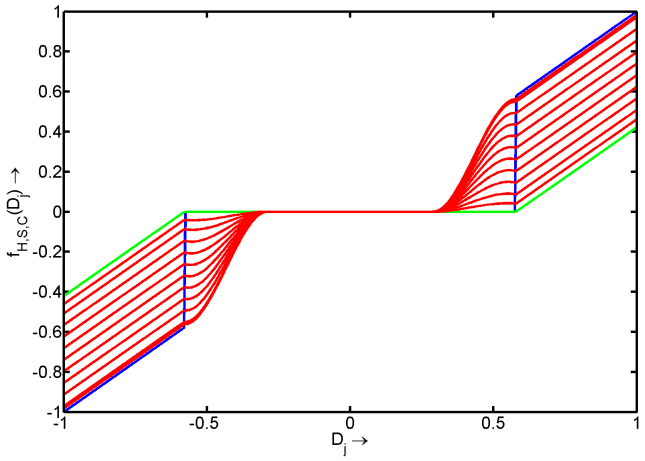


Figure 10: *Coefficients Thresholding: Hard-Thresholding (Blue), Soft-Thresholding (Green) and Custom-Thresholding (Red).*

3.2. Method II - DWT / DCT / LPC / Q

In this method, as depicted in Fig. 11, we present our transient encoding approach. To motivate this choice, we compare this method with various combinations of dyadic approximations applied to the detected transient. The discrete wavelet transform (DWT) followed respectively by discrete cosine transform (DCT), linear prediction coding (LPC) and quantization (Q) are applied on the detected transient. In the analysis part (see Fig. 11 upper), the

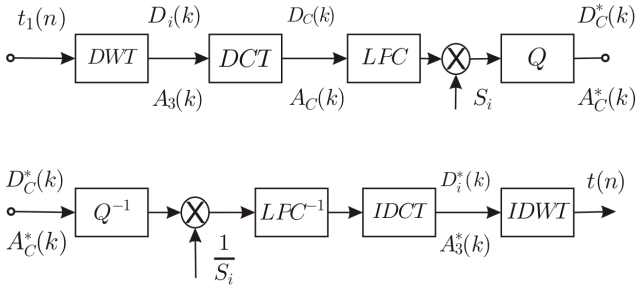


Figure 11: *Method II: Discrete Wavelet Transform (DWT), Discrete Cosine Transform (DCT), Linear Prediction Coding (LPC) and Quantization (Q).*

DWT coefficients are transformed with the discrete cosine transform (DCT-II) yielding

$$D_{C_j}(k) = \beta(k) \sum_{n=1}^{N} D_j(n) \cos(\frac{(2n+1)k\pi}{2N}) \qquad (13)$$

for the detail coefficients and

$$A_{C_j}(k) = \beta(k) \sum_{n=1}^{N} A_j(n) \cos(\frac{(2n+1)k\pi}{2N}) \qquad (14)$$

for the approximation coefficients, with

$$\beta(k) = \begin{cases} \sqrt{\frac{1}{N}} & \text{if } k = 1, \\ \sqrt{\frac{2}{N}} & \text{for } k = 2, .., N. \end{cases} \qquad (15)$$

where N is the length of the coefficient array in the considered subband and j is the index of the corresponding subband. On the DCT transformed coefficients linear prediction coding (LPC) is then performed. A prediction filter with model order $p = 8$ is used for this method. In each subband the prediction error is first normalized (S_i) before quantization is applied on it. The original 16 bit quantization word-length is here reduced to only 4 bit. In the synthesis stage, the inverse of the scaling factor ($1/S_i$) is first multiplied with the quantized signal before LPC synthesis filter is applied on it. Assuming that the analysis filter $A(z)$ used in the LPC part is the one designed with Eq. (3), we can expect perfect reconstruction of the coefficients using the inverse filter $H(z)$ (Eq. (4)). The inverse discrete cosine transform (IDCT) and inverse discrete wavelet transform (IDWT) are performed for final reconstruction.

3.3. Method III - DWT / LPC / Q

This method, as presented in Fig. 12, is a modified version of the previous method. In the analysis part, linear prediction cod-

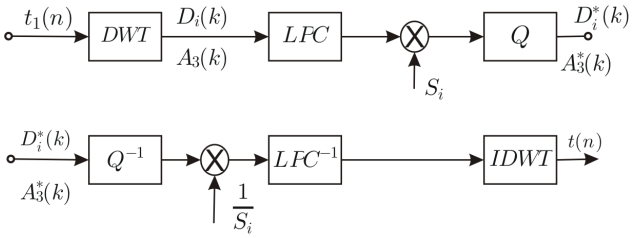


Figure 12: *Method III: Discrete Wavelet Transform (DWT), Linear Prediction Coding (LPC) and Quantization (Q).*

ing (LPC) is directly applied to the coefficients from the discrete wavelet transform (DWT). We finally quantize the prediction error using the same word-length like in method II. A prediction filter with model order $p = 8$ is again used here. The prediction error is normalized in each subband before quantization is applied. The normalized prediction error is then 4 bit quantized. In the synthesis stage, the inverse of the scaling factor ($1/S_i$) is multiplied with the quantized signal before LPC synthesis filter is applied on it. We finally reconstruct the signal the inverse discrete wavelet transform (IDWT).

3.4. Method IV - DCT / LPC / Q

The last method, as shown in Fig. 13, deals with the discrete cosine transform (DCT) directly applied to the detected transient signal. The DCT transformed signal is then linear predicted yielding a prediction error which will be finally quantized. The prediction error is first normalized in each subband before quantization is applied on it. The original 16 bit quantization word-length is again here reduced to only 4 bit. In the synthesis stage, the inverse of the scaling factor is first multiplied with the 4 bit quantized signal before LPC synthesis filter is applied on it. We finally reconstruct the signal using the inverse discrete cosine transform (IDCT).

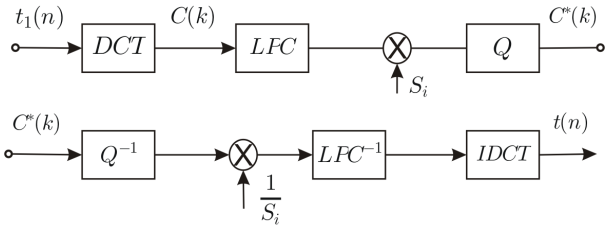


Figure 13: *Method IV: Discrete cosine Transform (DCT), Linear Prediction Coding (LPC) and Quantization (Q).*

4. SIMULATION RESULTS

We have applied the four methods to detected transient signal and have calculated the L2-norm

$$l = \sqrt{\sum_{n=1}^{N} r^2(n)} \tag{16}$$

of the residual signal from each method. In [16], a similar approach has been used for comparison of reconstruction results in image processing. We should point out that this way of characterizing the residual signal is purely numerical and does not take perceptual hearing considerations into account. Although there might be a correlation between numerical error and sound quality. In Fig. 15 results obtained when applying only the Spectral modelling Synthesis (SMS) to the castanet signal are shown. The original signal $x(n)$, the reconstructed signal $y(n)$ and the residual signal $r_{x-y}(n)$ with the L2-norm are presented. From Fig. 16 up to Fig. 19, results obtained with approaches depicted in Fig. 1 applied to the same castanet samples are shown. We can easily recognize that the reconstructed signal $y(n)$ is very close the original signal $x(n)$ for all the methods, with small differences regarding their L2-norm. We can also notice that the three components approach really outperforms the Spectral modelling Synthesis (SMS) approach for this kind of signal. We have applied the same simulation to different audio signals. For the sake of completeness, we will show here the results obtained with glockenspiel samples. Fig. 20 shows results obtained when applying only the Spectral modelling Synthesis (SMS) to the glockenspiel signal. In Fig. 21 results obtained when applying the SMS approach combined with the second transient encoding approach to the glockenspiel samples are presented. It is worth mentioning that the three components approach again outperforms the Spectral modelling Synthesis (SMS) approach regarding L2-norm. Similar results are also obtained applying the same approaches to samples from ABBA sound file (see Fig. 22). Improvement of the SMS approach is expected with the new promising method proposed [17]. This method analyzes the noise without any prior knowledge of the sinusoids model as presented in [2]. For all the methods, where linear prediction coding (LPC) is used, model order $p = 8$ has been applied. The original 16 bit quantization word-length of the input signal is reduced to only 4 bit for these methods. Regarding L2-norm results, we can notice that the three first methods remain close to each other. The L2-norm of residual signal obtained with method IV is bigger than the one obtained with other three methods. In Table 1 and Fig. 14, results obtained during listening test with headphones are presented. The subjects recruited for this test are all working in our lab. All the subjects assigned a grade of 100 to the hidden reference signal. While Subject F is the only one who graded the proposed method lower than the other methods, subject I even graded method III higher than others. Subject C did not perceive any difference between all the methods. But nevertheless method II is graded the best and method IV is scored lowest by all the subjects. With results presented in Table 1 and the L2-norm of the residual signal, a correlation between numerical error and sound quality is somehow observed.

Subject	Method.I	Method.II	Method.III	Method.IV
A	80	80	80	40
B	70	80	70	80
C	90	90	90	90
D	75	90	80	60
E	90	93	90	90
F	80	60	60	80
G	78	88	80	75
H	90	90	80	80
I	85	95	98	80
J	92	90	85	90
K	85	90	80	80
L	75	90	85	80
μ	82.5	86.33	81.5	77.08
σ	6.91	9.02	9.34	13.61
Interval	$\pm$ 4.59	$\pm$ 5.99	$\pm$ 6.20	$\pm$ 9.03

Table 1: Results from listening test using headphones. μ is the arithmetic average and σ is the standard deviation. **Interval**: 95 % confidence interval.

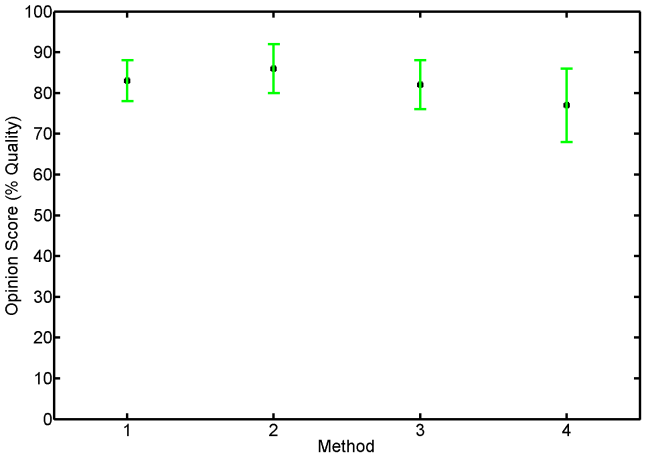


Figure 14: *Results from listening test using headphones. Bars denote 95 % confidence interval*

5. CONCLUSION

We have presented a method for transient detection based upon linear prediction combined with envelope estimation. This method succeeds in detecting transients in various kinds of audio signals. We have shown several alternatives for transient encoding using dyadic approximations. Regarding the L2-Norm of the residual signal, the method based on the discrete wavelet transform (DWT)

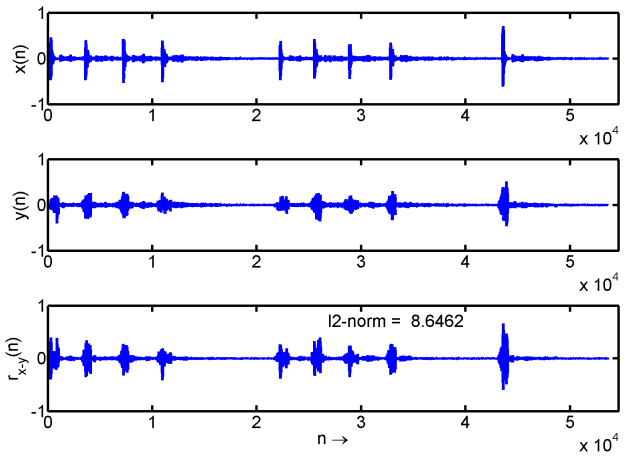


Figure 15: *Original signal x(n) (Castanets), SMS synthesized signal y(n), residual signal* $r_{x-y}(n)$.

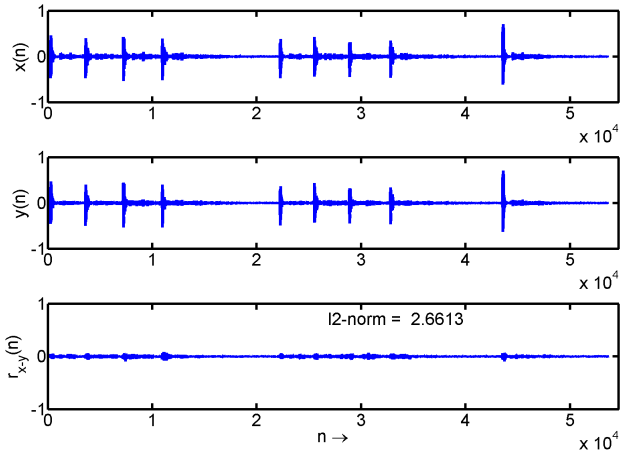


Figure 16: *Original signal x(n) (Castanets), SMS + Method I y(n), residual signal* $r_{x-y}(n)$.

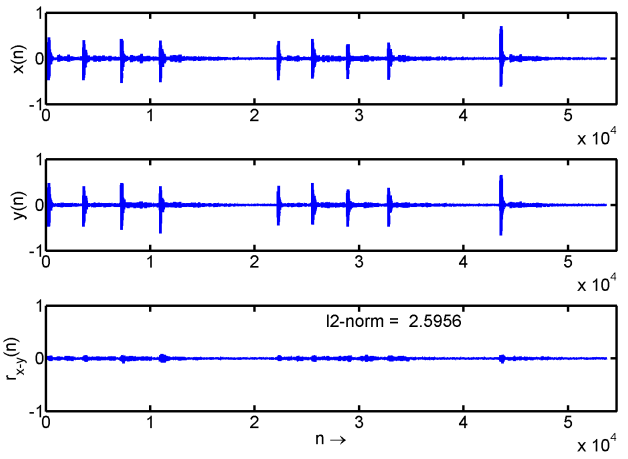


Figure 17: *Original signal x(n) (Castanets), SMS + Method II y(n), residual signal* $r_{x-y}(n)$.

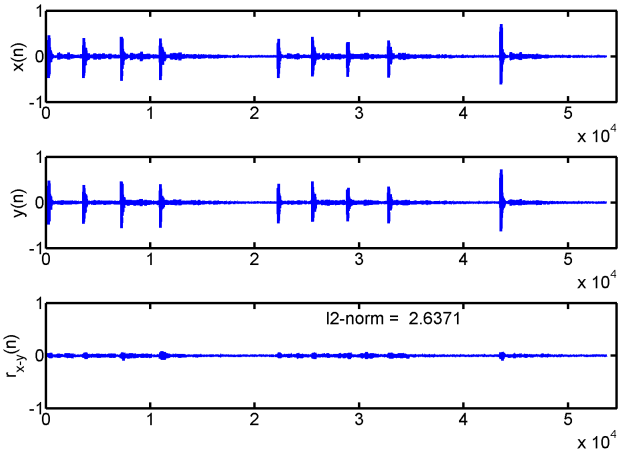


Figure 18: *Original signal x(n) (Castanets), SMS + Method III y(n), residual signal* $r_{x-y}(n)$.

followed by the discrete cosine transform (DCT), Linear Prediction Coding (LPC) and 4 bit quantization remains close to method I and III for all the tested signals. Subjective listening tests combined with the L2-Norm of the residual signal, indicate exactly that method II outperforms others. For future work, the same listening tests will be repeated in modified order and with many different audio files to observe if the trend remains the same.

6. REFERENCES

[1] R. McAulay and T. F. Quatieri. Speech analysis/synthesis based on a sinusoidal represenatation. *IEEE Transactions on Acoustics, Speech and Signal Processing, vol. 34, pp. 744-754*, 1986.

[2] X. Serra and J.O. Smith. Spectral modeling synthesis: A sound analysis/synthesis system based on a deterministic plus stochastic decomposition. In *Computer Music Journal, vol. 14(4), pp. 14-24*, 1990.

[3] M. Desainte-Catherine and S. Marchand. High precision fourier analysis of sounds using signal derivatives. In *J. Audio Eng. Soc. ,vol. 48 N° 7/8, pp. 654-667*, May 2000.

[4] T. Verma and T.H.Y. Meng. Extending spectral modeling synthesis with transient modeling synthesis. In *Computer Music Journal, vol. 24(2), pp. 47-59*, 2000.

[5] S. Molla L. Daudet and B. Torrésani. Transient detection and encoding using wavelet coefficient trees. In *Proc. of the GRETSI'01 conference, F. Flandrin Ed.*, 2001.

[6] Molla and B. Torrésani. Hidden markov tree based transient estimation for audio coding. In *Proc. of the conference ICME´02 conference, P.Vandergheynst Ed.*, 2002.

[7] M. Sandler C. Duxbury and M. Davies. A hybrid approach to music note onset detection. In *Proc. 5th International Conference on Digital Audio Effects (DAFx-02), Hamburg, Germany, pp. 33-38*, September 2002.

[8] J. Uscher. Extraction and removal of percussive sounds from musical recordings. In *Proc. 9th International Conference*

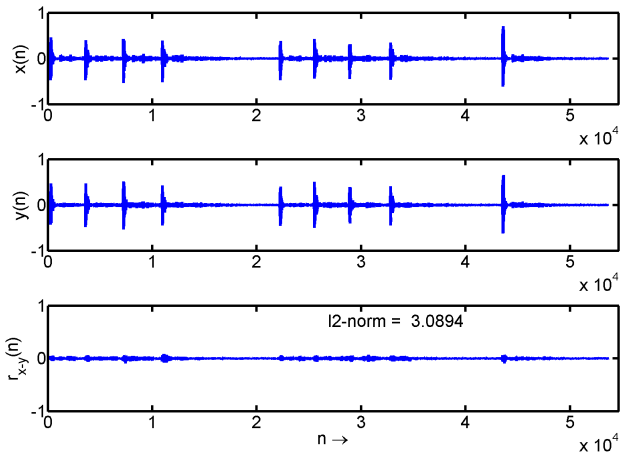


Figure 19: *Original signal x(n) (Castanets), SMS + Method IV y(n), residual signal* $r_{x-y}(n)$.

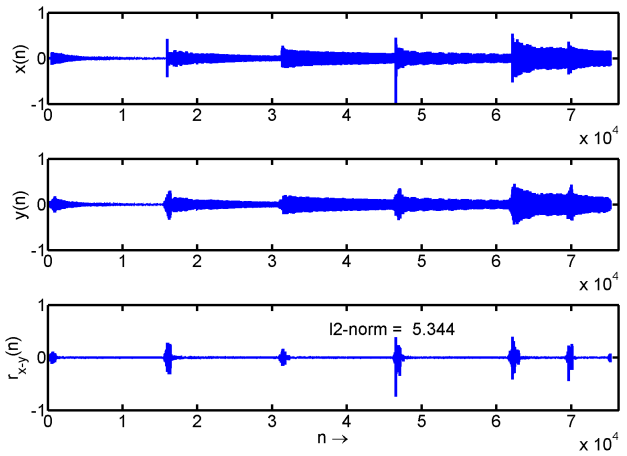


Figure 20: *Original signal x(n) (Glockenspiel), SMS synthesized signal y(n), residual signal* $r_{x-y}(n)$.

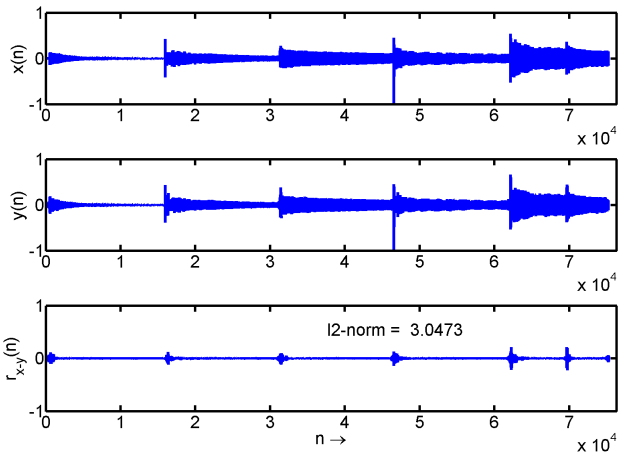


Figure 21: *Original signal x(n) (Glockenspiel), SMS + Method II y(n), residual signal* $r_{x-y}(n)$.

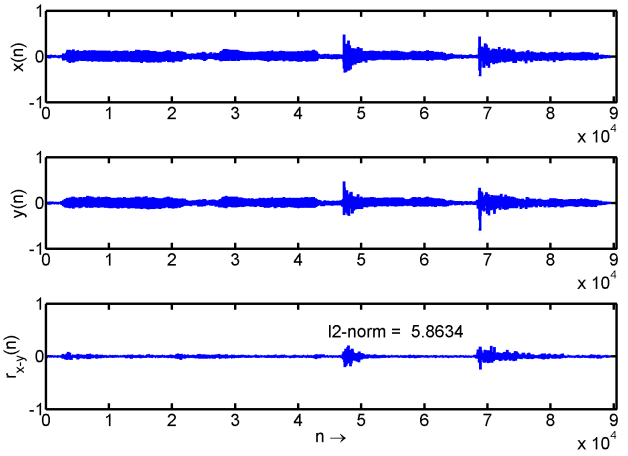


Figure 22: *Original signal x(n) (ABBA sound file), SMS + Method II y(n), residual signal* $r_{x-y}(n)$.

on Digital Audio Effects (DAFx-06), Montreal, Canada, pp. 263-266, September 2006.

[9] L. Daudet. A review on techniques for the extraction of transients in musical signals. In *Proc. of the CMMR´05 conference, Pisa, Italy, pp. 219-232*, 2005.

[10] M. Lagrange and S. Marchand. Real-time additive synthesis of sound by taking advantage of psychoacoustics. In *Proc. 4th International Conference on Digital Audio Effects (DAFx-01), Limerick, Ireland, pp. 5*, December 2001.

[11] I. Kauppinen and K. Roth. Audio signal extrapolation - theory and applications. In *Proc. 5th International Conference on Digital Audio Effects (DAFx-02), Hamburg, Germany, pp. 105-110*, September 2002.

[12] S. V. Vaseghi. *Advanced Signal Processing and Digital Noise Reduction*. Willey &. Teubner, Stuttgart, 1996. ISBN 3-519-06451-0.

[13] M. Davies C. Duxbury and M. Sandler. Separation of transient information in musical audio using multiresolution analysis techniques. In *Proc. 4th International Conference on Digital Audio Effects (DAFx-01), Limerick, Ireland, pp. 1*, December 2001.

[14] M. Sandler C. Duxbury, J. P. Bello and M. Davies. A comparison between fixed and multiresolution analysis for onset detectionin musical signals. In *Proc. 7th International Conference on Digital Audio Effects (DAFx-04), Naples, Italy, pp. 207-211*, October 2004.

[15] B. J. Yoon and P. P. Vaidyanathan. Wavelet-based denoising by customized thresholding. In *ICASSP, Montreal, Canada, vol. 2, pp. ii-925-8*, May 2004.

[16] L. Yaroslavsky and Y. Chernobrodov. Dct and dft discrete sinc-interpolation methods for direct fourier tomographic reconstruction. In *ISPA, Rome Italy, pp. 405*, 2003.

[17] P. Hanna G. Meurisse and S. MARCHAND. A new analysis method for sinusoids + noise spectral models. In *Proc. 9th International Conference on Digital Audio Effect (DAFx-06), Montreal, Canada*, September 2006.

ADJUSTABLE BOUNDARY CONDITIONS FOR MULTIDIMENSIONAL TRANSFER FUNCTION MODELS

Rudolf Rabenstein and Stefan Petrausch

Multimedia Communication and Signal Processing
University of Erlangen-Nuremberg, Cauerstr. 7, 91058 Erlangen, Germany
{stepe,rabe}@LNT.de

ABSTRACT

Block based physical modeling requires to provide a library of modeling blocks for standard components of real or virtual musical instruments. Complex synthesis models are built by connecting standard components in a physically meaningful way. These connections are investigated for modeling a resonating structure as a distributed parameter system. The dependence of a resonator's spectral structure on the termination of its ports is analyzed. It is shown that the boundary conditions of a distributed parameter system can be adjusted by proper termination only. Examples show the corresponding variation of the resonator's spectral structure in response to variations of the external termination.

1. INTRODUCTION

1.1. Block Based Physical Modeling

There is a rich set of tools available for digital sound synthesis: wavetable synthesis, frequency modulation (FM), additive and subtractive synthesis, granular and concatenative synthesis, and various flavors of physical modeling. Rather than generating more and more new synthesis methods, recent advances have focused on the combination of different synthesis methods. In the context of physical modeling, a methodology for the block-wise synthesis of virtual musical instruments has been developed under the name of *block based physical modeling*.

Block based modeling separates the tasks of *component design* and *model building*. Component design means that various components of real or virtual instruments like strings, membranes, air columns, piano hammers, mallets, etc. are modeled and implemented independently of each other. The resulting component models (the *blocks*) are stored in a block library for later use. Model building means to built a virtual instrument from its components by selecting the appropriate blocks from the library and to connect them in a meaningful way. An overview on methods and synthesis tools for block based modeling can be found in [1] and the literature cited there; a detailed account of the fundamentals is given in [2].

1.2. Signals and Ports

This procedure is well known from signal based simulation environments like SIMULINK or programming languages for audio signals like Pure Data (PD). The block structure in these implementations resembles signal flow graphs known from systems and control theory. Blocks for processing signals have well defined inputs and outputs. The connection of the output of one block to the input of the next one does not change the values of the output signal. This property is easy to implement in software but it is also shared by specialized hardware like analog modular synthesizers. In detail, electronic circuitry with low output impedance and high input impedance ensures that the output signal is not affected by connections to a limited number of inputs.

The situation is different when the blocks model physical components. At first, the related quantities (e.g. pressure and particle velocity in a pipe) are not per se given as input or output signals. Furthermore, connecting two blocks will affect all related quantities. This situation is usually described by so called ports, a combination of two or more variables like pressure and flow, force and velocity, or voltage and current. Connecting two physical modeling blocks means to connect the respective port variables, which in turn will change the behavior of both blocks.

1.3. Boundary Conditions

Designing blocks for physical modeling frequently requires to consider distributed parameter systems like strings, membranes, and air columns. Their implementation is based on a mathematical description in the form of partial differential equations (PDEs) and their respective boundary conditions. In musical instruments, boundary conditions are given e.g. by the fixing of a string, a membrane, or a plate, or by the termination of an air column. The type of boundary conditions determines the sound of a resonating structure, as is well known from string, brass, and woodwind instruments or from organ pipes.

Boundary conditions of distributed parameter systems are closely related to the port variables of their block implementations. In short, the port variables are the values of the block model at the interface to the outside world, i.e. to other block models. Conditions on the port variables imposed by block connections or terminations constitute the boundary conditions for the distributed parameter block. Examples are the excitation of a string, which is zero at a fixed end or the pressure in a pipe which is zero at an open end.

The mathematical literature classifies boundary conditions of the first, second, and third kind [3, 4]. Boundary conditions of the first and second kind prescribe the values of port variables or their derivatives. Boundary conditions of the third kind prescribe relations between the port variables. These relations may be real or complex valued and are given in terms of reflection factors or impedances. Methods for the investigation of resonance modes in a one-dimensional medium with two resistive boundaries have been compared in [5].

1.4. Connecting Blocks

At this point, the separation of component design and model building discussed above poses a problem which is the topic of this contribution: During component design, i.e. when a distributed parameter model is implemented, the boundary conditions for the use of this component for model building are not known. Moreover, a certain block has to work in a physically meaningful way in different kinds of connections.

However mathematical rigor requires that the boundary conditions are included in the definition of a distributed parameter model in order to constitute a properly posed problem. This means that a distributed parameter block at first has to be designed and implemented for a certain set of boundary conditions and later used in block connections which impose other boundary conditions.

This problem can also be expressed in musical terms. When a block model of e.g. a string with fixed ends is connected with another block, e.g. a sound board, will the spectral structure of the block model change accordingly? Is it sufficient to provide the correct port connections to the existing blocks or is it necessary to redesign the string model?

This problem is discussed here for a specific case. A block model for an air column with standard boundary conditions is terminated by an external component and the resulting spectral properties are investigated. The answer to the question above is given by formulating the problem as a feedback structure and by analyzing it in terms of basic control theory.

2. PROBLEM DESCRIPTION

This section describes the problem in general terms. Block models of distributed parameter systems are introduced, the boundary conditions are formulated, and an example for wave propagation is presented.

2.1. Block Models of Distributed Parameter Systems

A general distributed parameter system with one spatial dimension is shown in Fig. 1. It may represent a vibrating string, an air column, or another type of waveguide. The spatial coordinate is denoted by x, the model is defined within the one-dimensional spatial region $V = [0, l]$ with the boundary $\partial V = \{0; l\}$. For all boundary points $x_\mathrm{b} \in \partial V$, i.e. $x_\mathrm{b} \in \{0; l\}$, the behavior is determined by two physical variables $y_1(x_\mathrm{b})$ and $y_2(x_\mathrm{b})$. They constitute the port variables introduced above. These variables may represent force and deflection, pressure and particle velocity, or other pairs of across and through variables, depending on the nature of the distributed system. Two of these variables are sufficient to describe simple resonating structures. More involved models with more than two variables can be investigated in the same way, but they are not discussed here.

The internal behavior of the system in Fig. 1 is described in terms of a vector partial differential equation (1). The vector $\mathbf{y}(x,t)$ consists of the two variables $y_1(x,t)$ and $y_2(x,t)$, the vector $\mathbf{v}(x,t)$ describes a possible excitation function. The matrices $\mathbf{B}_1$ and $\mathbf{B}_2$ describe the partial differential operators in detail. An example is given in Sec. 2.3.

$$\left[\mathbf{B}_1 \frac{\partial}{\partial x} + \mathbf{B}_2 \frac{\partial}{\partial t}\right] \mathbf{y}(x,t) = \mathbf{v}(x,t) \;, \quad x \in V \tag{1}$$

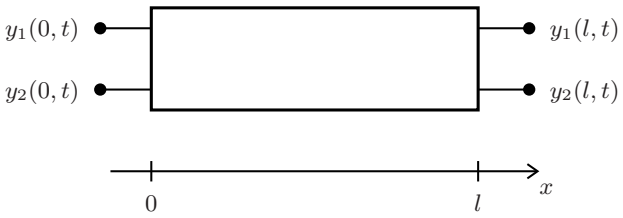


Figure 1: Sketch of a spatially one-dimensional block model. The boundary consists of two points $x_\mathrm{b} \in \{0; l\}$.

The port variables, i.e. the the outcome at the boundaries $\mathbf{y}(x,t)$ for $x = x_\mathrm{b}$ are given by

$$\mathbf{y}(0,t) = \begin{pmatrix} y_1(0,t) \\ y_2(0,t) \end{pmatrix} \;, \qquad \mathbf{y}(l,t) = \begin{pmatrix} y_1(l,t) \\ y_2(l,t) \end{pmatrix} \;. \tag{2}$$

2.2. Boundary Behavior

The port variables are neither inputs nor outputs in the sense that e.g. $y_1(x_\mathrm{b})$ is independent of $y_2(x_\mathrm{b})$ and $y_2(x_\mathrm{b})$ is determined only by $y_1(x_\mathrm{b})$. If the port variables $\mathbf{y}(x,t)$ for $x = x_\mathrm{b}$ are connected to the port variables of another block, then their values are determined by the interaction of both blocks.

This interaction happens instantly for continuous-time systems. For discrete-time systems, it is necessary to ensure computability by avoiding delay-free loops. To this end, the elements of $\mathbf{y}(x_\mathrm{b}, t)$ or combinations thereof have to be divided into input and output variables. The formal description of this division requires to introduce the normal component of the differential operator (see [2])

$$\mathbf{B}_\mathrm{n} = n_1 \mathbf{B}_1 + n_2 \mathbf{B}_2 \tag{3}$$

with the normal vector

$$\mathbf{n}_\mathrm{b} = \begin{pmatrix} n_1 \\ n_2 \end{pmatrix} \;. \tag{4}$$

This notation allows for rather general boundary conditions including time-varying boundaries. For the consideration of boundary conditions at time-invariant boundary points, $\mathbf{n}_\mathrm{b}$ has the values

$$\mathbf{n}_\mathrm{b} = \begin{pmatrix} -1 \\ 0 \end{pmatrix} \quad \text{for} \quad x = 0, \qquad \mathbf{n}_\mathrm{b} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \text{for} \quad x = l \;, \tag{5}$$

such that

$$\mathbf{B}_\mathrm{n} = \pm \mathbf{B}_1 \;. \tag{6}$$

Now the boundary inputs and boundary outputs can be defined by introducing the boundary input operator $\mathbf{f}_\mathrm{b}$ and the boundary output operator $\mathbf{f}_\mathrm{o}$, both are two-element column vectors. They define the input signal $v_\mathrm{b}(x_\mathrm{b})$ and the output signal $y_\mathrm{b}(x_\mathrm{b})$ at the boundary x_b as

$$\mathbf{f}_\mathrm{b}^\mathrm{H} \mathbf{B}_\mathrm{n} \mathbf{y}_\mathrm{b}(x_\mathrm{b}) = v_\mathrm{b}(x_\mathrm{b}) \qquad \text{input,} \tag{7a}$$

$$\mathbf{f}_\mathrm{o}^\mathrm{H} \mathbf{B}_\mathrm{n} \mathbf{y}_\mathrm{b}(x_\mathrm{b}) = y_\mathrm{b}(x_\mathrm{b}) \qquad \text{output .} \tag{7b}$$

The superscript H denotes the Hermitian vector or matrix. For real valued boundary operators, it is equal to the transposed vector.

The boundary input operator $\mathbf{f}_\mathrm{b}$ and the boundary output operator $\mathbf{f}_\mathrm{o}$ can be combined to a matrix representation to replace (7a,7b) by

$$\left(\begin{array}{c:c} \mathbf{f}_\mathrm{b} & \mathbf{f}_\mathrm{o} \end{array} \right)^\mathrm{H} \mathbf{B}_\mathrm{n} \mathbf{y}(x_\mathrm{b}) = \left(\begin{array}{c} v_\mathrm{b}(x_\mathrm{b}) \\ y_\mathrm{b}(x_\mathrm{b}) \end{array} \right) . \tag{8}$$

This definition of input and output values is only meaningful if $v_\mathrm{b}(x_\mathrm{b})$ and $y_\mathrm{b}(x_\mathrm{b})$ are not identical, i.e. if

$$\mathrm{rank}\left\{ \left(\begin{array}{c:c} \mathbf{f}_\mathrm{b} & \mathbf{f}_\mathrm{o} \end{array} \right)^\mathrm{H} \mathbf{B}_\mathrm{n} \right\} = \mathrm{rank}\left\{ \mathbf{B}_\mathrm{n} \right\} . \tag{9}$$

Fig. 2 shows the relation (8) between the port variables at $x = l$ and the input and output signals $v_\mathrm{b}(l,t)$ and $y_\mathrm{b}(l,t)$. The assignment between the port variables and the input and output signals is defined by the boundary input operator $\mathbf{f}_\mathrm{b}$ and the boundary output operator $\mathbf{f}_\mathrm{o}$. Only the port at $x = l$ is considered here. Similar results hold also for the port at $x = 0$ with the appropiate normal vector from (5).

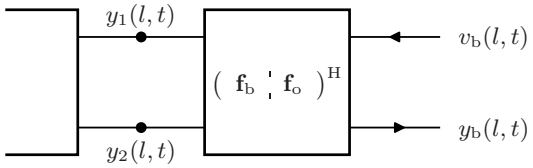


Figure 2: Spatial one-dimensional model with a specific boundary behavior described by equation (8).

2.3. Example: Wave Equation

As an example serves an air column with sound pressure $p = p(x,t)$, particle velocity $v = v(x,t)$, mass density ϱ_0, and speed of sound c. A simple distributed parameter model is given by

$$\left[\underbrace{\mathbf{I}_0}_{\mathbf{B}_1} \frac{\partial}{\partial x} + \underbrace{\left(\begin{array}{cc} 0 & -1 \\ -1/c^2 & 0 \end{array} \right)}_{\mathbf{B}_2} \frac{\partial}{\partial t} \right] \underbrace{\left(\begin{array}{c} p \\ -\varrho_0 v \end{array} \right)}_{\mathbf{y}} = \underbrace{\left(\begin{array}{c} 0 \\ -f_\mathrm{e} \end{array} \right)}_{\mathbf{v}} \tag{10}$$

where the underbraces indicate the corresponding vectors and matrices of the general system (1).

The normal component of the differential operator according to (6) is given by

$$\mathbf{B}_\mathrm{n} = \pm \mathbf{B}_1 = \pm \mathbf{I}_0 , \tag{11}$$

where $\mathbf{I}_0$ denotes the identity matrix.

Meaningful boundary input and output operators have to satisfy the rank condition

$$\mathrm{rank}\left\{ \left(\begin{array}{c:c} \mathbf{f}_\mathrm{b} & \mathbf{f}_\mathrm{o} \end{array} \right) \right\} = \mathrm{rank}\left\{ \mathbf{I}_0 \right\} = 2 \tag{12}$$

If the case of hard reflecting walls on both sides (i.e. at $x = 0$ and $x = l$) is considered as boundary condition, then the particle velocity at the boundary must be zero. If (10) is regarded as a PDE in terms of the sound pressure then these boundary conditions are of the second kind (Neumann boundary conditions). This assumption implies together with the rank condition (12)

$$\left(\begin{array}{c:c} \mathbf{f}_\mathrm{b2} & \mathbf{f}_\mathrm{o2} \end{array} \right) = \left(\begin{array}{cc} 0 & 1 \\ 1 & 0 \end{array} \right) , \tag{13}$$

and thus the following assignment between port variables and the input and output signals

$$\left(\begin{array}{cc} 0 & 1 \\ 1 & 0 \end{array} \right) \left(\begin{array}{c} y_1(l,t) \\ y_2(l,t) \end{array} \right) = \left(\begin{array}{c} v_\mathrm{b2}(l,t) \\ y_\mathrm{b2}(l,t) \end{array} \right) . \tag{14}$$

The subscripts $\mathrm{b}2$ and $\mathrm{o}2$ denote boundary conditions of the second kind. The relation (14) is shown graphically in Fig. 3 by specializing the boundary operators from Fig. 2 to (13). Another type of boundary conditions is considered in the following section.

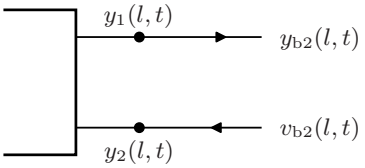


Figure 3: Block model with boundary conditions of the second kind.

3. ADJUSTABLE BOUNDARY CONDITIONS

This section describes a different kind of boundary conditions which include adjustable parameters. If not connected to another block, this kind of boundary conditions is equivalent to a port termination.

3.1. Boundary Conditions of the Third Kind

The assignment of boundary conditions of the second kind (14) declares one port signal ($y_1(l,t)$) to be the output ($y_\mathrm{b2}(l,t)$) and the other port signal ($y_2(l,t)$) to be the input ($v_\mathrm{b2}(l,t)$). But it is also possible to declare a certain linear combination of the port signals to be the output and another linear combination to be the input. Such an assignment corresponds to boundary conditions of the third kind (also called Robin's boundary conditions) and is given by

$$\left(\begin{array}{c:c} \mathbf{f}_\mathrm{b} & \mathbf{f}_\mathrm{o} \end{array} \right) = \left(\begin{array}{cc} 1 & g_2 \\ g_1^{-1} & 1 \end{array} \right) \qquad \text{with } g_1 \neq g_2 , \tag{15}$$

where g_1 and g_2 are real admittances. Their physical dimensions have to be compatible with the port variables $y_1(l,t)$ and $y_2(l,t)$. The rank condition (12) requires $g_1 \neq g_2$.

The input and output variables $v_\mathrm{b3}(l,t)$ and $y_\mathrm{b3}(l,t)$ are assigned by

$$\left(\begin{array}{cc} 1 & g_1^{-1} \\ g_2 & 1 \end{array} \right) \left(\begin{array}{c} y_1(l,t) \\ y_2(l,t) \end{array} \right) = \left(\begin{array}{c} v_\mathrm{b3}(l,t) \\ y_\mathrm{b3}(l,t) \end{array} \right) . \tag{16}$$

The subscripts $\mathrm{b}3$ and $\mathrm{o}3$ denote boundary conditions of the third kind.

Due to the definition of input and output variables by (16), the port variables $y_1(l,t)$ and $y_2(l,t)$ themselves are not computed in a specific order. However, to establish a relation with boundary conditions of the second kind, now consider arbitrarily $y_1(l,t)$ as an output variable of the port at $x = l$. To realize the boundary block from Figure 2, one has to solve for $y_2(l,t)$ and $y_\mathrm{b3}(l,t)$

$$y_2(l,t) = g_1 \cdot v_\mathrm{b3}(l,t) - g_1 \cdot y_1(l,t) , \tag{17a}$$

$$y_\mathrm{b3}(l,t) = 1 \cdot y_2(l,t) + g_2 \cdot y_1(l,t) . \tag{17b}$$

Fig. 4 shows the lattice structure of the corresponding signal flow graph.

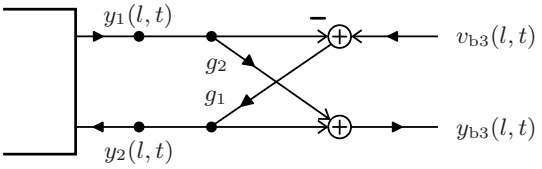


Figure 4: Block model with boundary conditions of the third kind.

3.2. Port Termination

Here no connection to other blocks is considered and therefore the input signal $v_{\mathrm{b}3}(l,t)$ is zero and the output signal $y_{\mathrm{b}3}(l,t)$ is not required. Then the realization of boundary conditions of the third kind simplifies to a termination of the port at $x = l$ with boundary conditions of the second kind (see Fig.3) by a negative admittance g_1 as shown in Fig. 5. Adjusting the coefficient $-g_1$ changes the character of this termination.

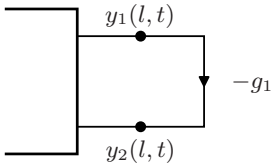


Figure 5: Simplified boundary conditions, the boundary input is zero and the boundary output is not required.

3.3. Realization of Adjustable Boundary Conditions

Based on the results of Sec. 3.1 and 3.2, adjustable boundary conditions with a parameter g_1 may be realized by block based physical modeling as follows:

- Design a standard block model according to Sec. 2.3 with boundary conditions of second kind. According to Fig. 2 one port variable is the input variable and the other one is the output variable. This block can be designed during component design and stored in a block library for later use.
- For realization of adjustable boundary conditions during model building, use the previously designed block and apply an external termination to its port at $x = l$. It consists of a feedback of the scaled port output back to the port input. The scaling multiplier may be any real number. According to Fig. 4 and Fig. 5 this procedure realizes boundary conditions of the third kind from an existing block model.

4. INTERPRETATION

Although the results of the previous section follow directly from the boundary conditions from equation (17), they do not explain the resulting change in the spectral structure of the initial block model. This section gives an interpretation of the previous results based on tools from basic control theory, i.e. transfer function formulation and feedback analysis.

4.1. Transfer Function Formulation

Now apply the procedure from Sec. 3.3 for a discrete-time block model:

- design a discrete-time model with boundary conditions of second kind.
- turn it into a model with boundary conditions of third kind by termination with a suitable admittance.

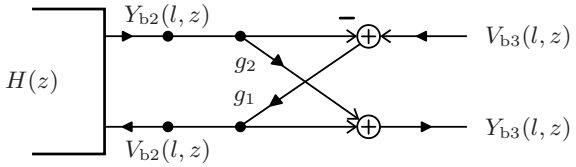


Figure 6: Block model with boundary conditions of third kind.

Fig. 6 shows the discrete-time version of Fig. 4. $V_{\mathrm{b}2}(l,z)$ and $Y_{\mathrm{b}2}(l,z)$ are z-transforms of discrete-time approximations $v_{\mathrm{b}2}[l,k]$ and $y_{\mathrm{b}2}[l,k]$ of their continuous counterparts $v_{\mathrm{b}2}(l,t)$ and $y_{\mathrm{b}2}(l,t)$. Corresponding relations hold for $V_{\mathrm{b}3}(l,z)$ and $Y_{\mathrm{b}3}(l,z)$. $H(z)$ is the transfer function of a discrete approximation of the distributed system with boundary conditions of the second kind with respect to its port variables, i.e.

$$H(z) = \frac{Y_{\mathrm{b}2}(l,z)}{V_{\mathrm{b}2}(l,z)} = \frac{N(z)}{D(z)} \,. \tag{18}$$

$N(z)$ and $D(z)$ denote the numerator and the denominator polynomial, respectively.

The form of $H(z)$ depends on the kind of discrete-time approximation of the distributed system. An example for a realization with the functional transformation method is given in Section 4.2. Other physical modeling methods like e.g. the digital waveguide method [6, 7] yield different transfer functions with similar behavior.

4.2. Example: Functional Transformation Method

The functional transformation method (FTM) is used for physical modeling digital sound synthesis of resonating structures like strings, bars, air columns, membranes, plates, and alike. It starts from a PDE description of the continuous-time, continuous-space model and derives a discrete-time model in the form of a parallel arrangement of simple transfer functions. More details on the procedure and examples for the use of multidimensional transfer function models can be found in [2, 8].

Here it is sufficient to show the complex representation of a typical configuration in Fig. 7. It consists of a parallel arrangement of first order systems with complex feedback coefficient a_μ and further multipliers b_μ and c_μ derived from certain eigenvalue problems [2, 8]. These eigenvalue problems (so-called Sturm-Liouville problems) consider the boundary conditions of the system, here boundary conditions of the second kind.

The transfer function of a single first order system is

$$H_\mu(z) = b_\mu c_\mu \frac{z}{z - a_\mu} = \frac{d_\mu z}{z - a_\mu} \quad \text{with } d_\mu = b_\mu c_\mu \,. \tag{19}$$

The complete transfer function is given by the sum of all N first order systems from (19) as (see Fig. 7)

$$H(z) = \sum_{\mu=1}^{N} H_\mu(z) = \sum_{\mu=1}^{N} \frac{d_\mu z}{z - a_\mu} \,. \tag{20}$$

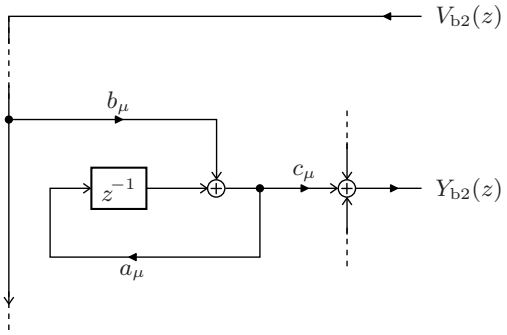


Figure 7: Structure of the discrete-time implementation achieved with the FTM, see for instance [8]. A finite number of N first order transfer functions with complex feedback coefficient a_μ are arranged in parallel.

The denominator $D(z)$ of (20) is the product of the denominators $z - a_\mu$ of the first order systems (19)

$$D(z) = \prod_{\nu=1}^{N} (z - a_\nu) \; . \tag{21}$$

The transfer function $H(z)$ from (20) turns with

$$D_\mu(z) = \frac{D(z)}{(z - a_\mu)} = \prod_{\substack{\nu=1 \\ \nu \neq \mu}}^{N} (z - a_\nu) \tag{22}$$

into

$$H(z) = \sum_{\mu} \frac{d_\mu z D_\mu(z)}{(z - a_\mu) D_\mu(z)} = \frac{N(z)}{D(z)} \; , \tag{23}$$

with the numerator

$$N(z) = z \sum_{\mu} d_\mu D_\mu(z) \; . \tag{24}$$

4.3. Feedback Analysis

In Sec. 3 it has been shown that boundary conditions of the third kind can be realized by external termination of a system with boundary conditions of the second kind. Now the relation between the transfer functions

$$H(z) = \frac{Y_{\mathrm{b}2}(l,z)}{V_{\mathrm{b}2}(l,z)} = \frac{N(z)}{D(z)} \; , \quad H_{\mathrm{b}3}(z) = \frac{Y_{\mathrm{b}3}(l,z)}{V_{\mathrm{b}3}(l,z)} = \frac{N_{\mathrm{b}3}(z)}{D_{\mathrm{b}3}(z)} \; . \tag{25}$$

is derived, where $V_{\mathrm{b}2}(l,z)$, $Y_{\mathrm{b}2}(l,z)$ and $V_{\mathrm{b}3}(l,z)$, $Y_{\mathrm{b}3}(l,z)$ correspond to Fig. 6. Contrary to the example in Sec. 4.2 no specific implementation of $H(z)$ is assumed.

The lattice structure from Fig. 6 is represented in matrix notation by (see (14) and (16))

$$\begin{pmatrix} 1 & g_1^{-1} \\ g_2 & 1 \end{pmatrix} \begin{pmatrix} Y_{\mathrm{b}2}(l,z) \\ V_{\mathrm{b}2}(l,z) \end{pmatrix} = \begin{pmatrix} Y_{\mathrm{b}3}(l,z) \\ V_{\mathrm{b}3}(l,z) \end{pmatrix} \tag{26}$$

or

$$Y_{\mathrm{b}2}(l,z) + g_1^{-1} V_{\mathrm{b}2}(l,z) = V_{\mathrm{b}3}(l,z) \tag{27}$$

$$g_2 Y_{\mathrm{b}2}(l,z) + V_{\mathrm{b}2}(l,z) = Y_{\mathrm{b}3}(l,z) \; . \tag{28}$$

The transfer function $H_{\mathrm{b}3}(z)$ follows from the division of (28) by (27) as (some arguments are omitted for convenience)

$$H_{\mathrm{b}3}(z) = \frac{Y_{\mathrm{b}3}}{V_{\mathrm{b}3}} = \frac{g_2 Y_{\mathrm{b}2} + V_{\mathrm{b}2}}{Y_{\mathrm{b}2} + g_1^{-1} V_{\mathrm{b}2}} \; . \tag{29}$$

Dividing by $V_{\mathrm{b}2}$ and using (18) gives

$$H_{\mathrm{b}3}(z) = \frac{g_2 H(z) + 1}{H(z) + g_1^{-1}} = \frac{g_2 \frac{N(z)}{D(z)} + 1}{\frac{N(z)}{D(z)} + g_1^{-1}} \; . \tag{30}$$

Multiplying by $D(z)$ results in

$$H_{\mathrm{b}3}(z) = \frac{g_2 N(z) + D(z)}{N(z) + g_1^{-1} D(z)} \; . \tag{31}$$

Multiplication with g_1 finally gives

$$H_{\mathrm{b}3}(z) = g_1 \frac{D(z) + g_2 N(z)}{D(z) + g_1 N(z)} = \frac{N_{\mathrm{b}3}(z)}{D_{\mathrm{b}3}(z)} \; . \tag{32}$$

Thus the transfer function for boundary conditions of the third kind $H_{\mathrm{b}3}(z)$ is expressed by the numerator $N(z)$ and denominator $D(z)$ of the transfer function for boundary conditions of the second kind.

Investigating the denominator of $H_{\mathrm{b}3}(z)$ yields the interpretation of the spectral effect of the external termination. Obviously, the resonances of $H_{\mathrm{b}3}(z)$ are given by the poles of the denominator polynomial $D_{\mathrm{b}3}(z)$

$$D_{\mathrm{b}3}(z) = D(z) + g_1 N(z) \; . \tag{33}$$

Therefore the real coefficient g_1 allows to shift the pole locations according to the boundary conditions of third kind in equation (15). The effect of the external termination is shown by examples in the next section.

5. RESULTS

For a more intuitive illustration of the results from the previous sections, the modeling scenario as depicted in Fig. 8 is applied. The underlying PDE is the wave equation as given in (10). The input of the model is $y_2(0,t)$ while it is solved for the output variable $y_1(l,t)$. According to the definition of the vectorial outcome $\mathbf{y}(x,t)$ in (10) this corresponds to the particle velocity at the left side as the input and the pressure at the right side for output.

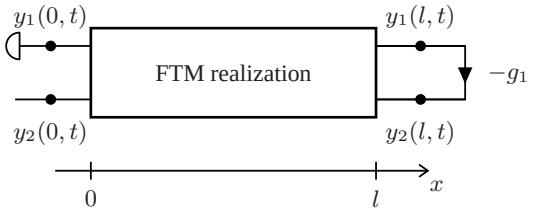


Figure 8: Sketch of the complete spatially one-dimensional block model. The boundary consists of two points $x_{\mathrm{b}} \in \{0; l\}$.

The boundary conditions at the left side are of the second kind, while the boundary conditions on the right side are adjustable via the parameter g_1. As well known from basic acoustics [9, 10],

boundary conditions of third kind (as given in (15)) directly correspond to a specific reflection factor α. In this scenario, this factor can be calculated by

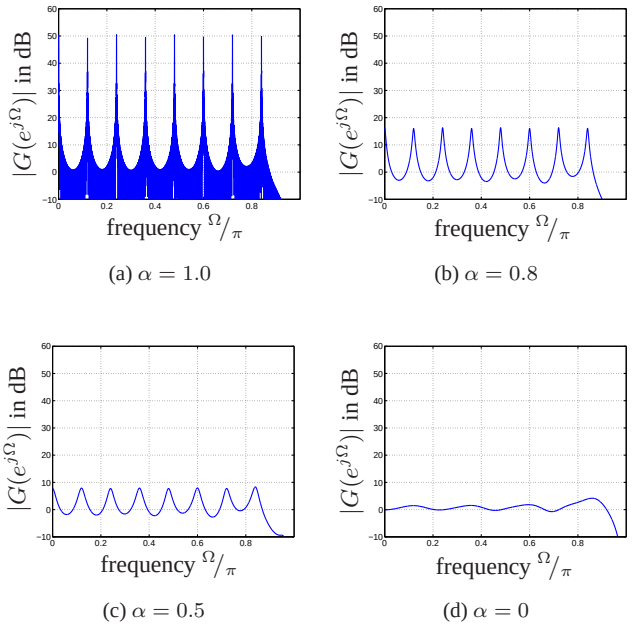
$$\alpha = \frac{1 + cg_1}{1 - cg_1} \ , \tag{34}$$

where c is the speed of sound in the medium. A discrete realization of the model is achieved with the FTM (see [8] for instance). The sampling period T is chosen to result in $N = 17$ first order systems, one with a_μ equal to zero and eight pairs of complex conjugate systems. The resulting discrete time transfer function

$$G(z) = \frac{Y_1(l, z)}{Y_2(0, z)} \tag{35}$$

is depicted in Fig. 9 for four different reflection coefficients. The transfer function in Fig. 9(a) represents boundary conditions of the second kind at $x = l$. The model obviously represents a comb filter what is the expected behavior, as the traveling waves are perfectly reflected at all boundaries. The other extremum is depicted in Fig. 9(d), where the reflection coefficient α is zero. The right side absorbs all incoming waves as good as possible, such that the impulse response from $y_2(0, t)$ to $y_1(l, t)$ is a simple Dirac impulse. The de facto transfer function in Fig. 9(d) however is not constant for all frequencies, as only 17 modes are considered in the simulation. Figs. 9(b) and 9(c) show intermediate cases for $\alpha = 0.8$ and $\alpha = 0.5$.

Obviously there is a gradual variation of the spectral properties of the system in Fig. 8, although the air column model itself does not change. The variations of the transfer function $G(e^{j\Omega})$ are only caused by adjusting the feedback coefficient g_1.



(a) $\alpha = 1.0$ (b) $\alpha = 0.8$

(c) $\alpha = 0.5$ (d) $\alpha = 0$

Figure 9: Normalized transfer function $G(z)$ from (34) evaluated at the unit circle $z = e^{j\Omega}$. The transfer function is depicted for different reflection coefficients α, which directly result from the feedback coefficient g_1 through (35).

6. CONCLUSIONS

The consideration of a simple example showed that models with adjustable boundary conditions can be built from multidimensional transfer functions with fixed boundary conditions. This result is important for the practical application of physical modeling. It is feasible to design components of musical instruments from the physical description of resonating structures. Although their implementation requires the assumption of fixed boundary conditions for a correct mathematical description, the spectral properties can be adjusted through proper port termination or, more general, by suitable connection to other modeling blocks.

These results have been obtained by considering one-dimensional transfer functions between suitably chosen port variables. The technique of calculating the denominator polynomial of a feedback structure from the corresponding open-loop transfer function is well-known in control theory as the root locus method. Considering the modeling block of a resonating structure with fixed boundary conditions as an open loop system (in control terms the *plant*), it is possible to adjust the properties of the closed loop system only by variations in the feedback path (the *controller*).

7. REFERENCES

[1] Rudolf Rabenstein, Stefan Petrausch, Augusto Sarti, Giovanni De Sanctis, Cumhur Erkut, and Matti Karjalainen, "Block based physical modeling for digital sound synthesis," *IEEE Signal Processing Magazine*, Mar. 2007.

[2] Stefan Petrausch, *Block Based Physical Modeling*, Ph.D. thesis, University of Erlangen-Nuremberg, 2007.

[3] Lokenath Debnath, *Nonlinear Partial Differential Equations for Scientists and Engineers*, Birkhäuser, Boston, MA, USA, 1997.

[4] Alan Jeffrey, *Applied Partial Differential Equations*, Academic Press, an Imprint of Elsevier Science, San Diego, CA, USA, 2003.

[5] Jean Kergomard, Vincent Debut, and Denis Matignon, "Resonance modes in a one-dimensional medium with two purely resisitve boundaries: Calculation methods, orthogonality, and completeness," *J. Acoustical Society of America*, vol. 119, no. 3, pp. 1356–1367, March 2006.

[6] Julius O. Smith, "Physical modeling using digital waveguides," *Computer Music Journal*, vol. 16, no. 4, pp. 74–91, 1992, Special Issue on Physical Modeling of Musical Instruments, Part I.

[7] Matti Karjalainen, Vesa Välimäki, and Tero Tolonen, "Plucked-string models: From the Karplus-Strong algorithm to digital waveguides and beyond," *Computer Music J.*, vol. 22, no. 3, pp. 17–32, 1998.

[8] Lutz Trautmann and Rudolf Rabenstein, *Digital Sound Synthesis by Physical Modeling using the Functional Transformation Method*, Kluwer Academic Publishers, New York, USA, 2003.

[9] Malcolm J. Crocker, Ed., *Handbook of Acoustics*, John Wiley & Sons, Inc., New York, USA, 1998.

[10] David T. Blackstock, *Fundamentals of Physical Acoustics*, John Wiley & Sons, Inc., New York, USA, 2000.

SELF-SUSTAINED VIBRATING STRUCTURES PHYSICAL MODELLING BY MEANS OF MASS-INTERACTION NETWORKS

François Poyer

ICA Laboratory
INPG, 46, avenue Félix Viallet
38031 Grenoble, France
Francois.Poyer@imag.fr

Claude Cadoz

ACROE & ICA Laboratory
INPG, 46, avenue Félix Viallet
38031 Grenoble, France
Claude.Cadoz@imag.fr

ABSTRACT

GENESIS is a sound synthesis and musical creation environment based on the mass-interaction CORDIS-ANIMA physical modelling formalism. It has got the noteworthy property that it allows to work both on sound itself and on musical composition in a single coherent environment. In this paper we present the first results of a study that is carried out with GENESIS on a particular type of models: self-sustained oscillating structures. By trying to build physical models of real instruments like bowed strings or woodwinds, our aim is to develop and analyse generic tools that can be used for the production of self-sustained oscillations on every mass-interaction network built with GENESIS. But, if the family of the self-sustained oscillating structures is very interesting to create rich timbres, it can also play a new and fundamental role at the level of the temporal macrostructure of the music (that of the gesture and the instrumental performance, as well as the composition). Indeed, it is possible, as we will propose in this paper, to use the relatively complex motion of a bowed macrostructure in a musical composition way, as a musical events generator.

1. INTRODUCTION

One of the reasons that motivated the introduction of sound synthesis by physical modelling was the search -for a better realism- of a naturalness of synthesized sounds. Logically researches began not on the sound itself, but on what produces this sound, that is the physical object, which is able to vibrate at acoustical frequencies. Indeed, human's ear was built by evolution for a precise purpose: to give us information about our environment. So, it is very sensitive to sounds (musical or not) produced by a well-determined physical cause. As a consequence, physical modeling will be an easier way to produce realistic sounds than signal processing.

But if we talk about music, what is physical is not only the sound produced by real instruments but also the instrumentalist's performance. Hence the use of physical modelling only to produce sounds with realistic timbre is a little restrictive. Using the physical modelling we can try to model also the instrumentalist itself, or at least some of its physical behaviour. This gives an approach of the sound construction at the scale of the musical macrostructure and, then offers a way to work at the compositional level.

GENESIS [1], a software based on mass-interaction modelling, takes this idea into account by proposing an environment where we can build objects that move at acoustical frequencies as well as at gesture frequencies (more generally at macrotemporal frequencies). As a result, within this environment, the arbitrary boundary between the timbre, the composition and the perfomance tends to be erased.

Among the infinity variety of physical models the environment allows to build, the specific category of self-sustained oscillating structures is particularly interesting. Indeed they allow to produce rich timbres but also, when used at low (gestural) frequencies, complex movements that can support rich expressivity. This article presents a study on this category of physical models which aims in developing simple models of, for example, violin, clarinet or oboe in the GENESIS environment and to find the relevant parameters of these models that can be used for rich timbre sound synthesis or for complex musical structures production.

After an introduction to the CORDIS-ANIMA formalism and the GENESIS environment, methods for self-sustained oscillating structures physical modelling will be presented on particular examples: bowed strings and woodwind instruments. The discussion will end on the large possibilities, for sound synthesis and for musical composition, enabled by this category of models.

2. PHYSICAL MODELLING WITH GENESIS

2.1. The theoretical basis of GENESIS: CORDIS-ANIMA

GENESIS is a coherent environment used for sound synthesis and more generally music creation. It is based on an axiomatic mass-interaction formalism called CORDIS-ANIMA [2]. Every object built with this formalism is constituted of different modules communicating with each other. We can distinguish two types of modules:

- <MAT> modules represent material points that for example may be provided with inertia.
- <LIA> modules link two <MAT> modules and represent the interactions between them (stiffness, viscous friction…)

Behind each module is an algorithm that calculates output variables according to input ones. For example, at each step, the algorithm of the <LIA> element called RES (which represents stiffness) takes as input the positions X_1 and X_2 of the two <MAT> elements that it links together. Then it gives as output the force that must be applied on the two <MAT> elements and which

modulus is $K|X_1-X_2|$ (where K is the stiffness coefficient). The <MAT> element called MAS, representing an ideal inertia, computes its position in time according to the force it receives as input.

So, with the CORDIS-ANIMA language, we can build an infinite variety of mass-interaction networks that correspond in a certain way to a space and time discrete view of Newton's laws. The main advantage with this coherent modular language is that everything is modelled with the same tools (the elementary modules), ensuring the consistency of every model. Furthermore, it is very simple to build interactions between two models developed with CORDIS-ANIMA, since they are done like interactions between two elementary <MAT> modules. Hence, it is possible to build complex models that are composed of many elements (for example the model of a string or of a pipe…) and simply make them interact by means of one or several <LIA> modules.

The CORDIS-ANIMA formalism is used to simulate physical objects we can see, hear or feel moving or vibrating, by using transducers. Different softwares based on the CORDIS-ANIMA formalism and dealing with image animation, sound or haptic perception have been developed. GENESIS is one of them, that is used for sound synthesis and music creation. It enables to build graphically any mass-interaction network with the <MAT> and <LIA> elementary modules. Here are basic modules used in the GENESIS environment:

- <MAT> modules: the SOL (fixed point), the MAS (ideal inertia), the CEL (one degree of freedom damped oscillator)
- <LIA> modules: the RES (stiffness), the FRO (viscous friction), the REF (viscoelastic link) and non-linear modules BUT and LNL (they will be developed below).
- <MAT> and <LIA> degenerated modules link the environment with external elements (loudspeaker, datafiles…): the ENF and the ENX (respectively force and position input), the SOF and the SOX (respectively force and position output). SOF and SOX modules are used to "hear" a structure vibrating.

The BUT is a viscoelastic conditional link, that is to say, a viscoelastic link which is effective if the difference between the positions of the two <MAT> elements that it links is under a given threshold. This module is often used for collision simulation.

The LNL module let us draw the interaction between two <MAT> by means of a function $F(\Delta X)$ or $F(\Delta V)$, with F the output force, ΔX and ΔV respectively the difference between positions or velocities of the two linked <MAT>. The user can draw every one-variable function he wants.

2.2. Time discretisation problems

Time discretisation implies working with recurrence relations instead of differential equations to calculate the model behaviour. Consequently, there are only some model parameters values that lead to convergent sequences. For example, for the elementary oscillator CEL which parameters are the inertia M, the stiffness K and the viscosity Z, the conditions of convergence are:

$$\begin{aligned} 0 \le Z \le M \\ 0 \le K + 2Z \le 4M \end{aligned} \quad (1)$$

These conditions calculated just for the elementary oscillator can nonetheless give a good idea of the convergence conditions for a more complex model. A general first approximation rule is hence that the masses of a model must be linked with <LIA> modules in which K and Z parameters must be smaller than the parameter M.

2.3. The Instrumentarium

In parallel to the GENESIS models development, a library of these models, called the *Instrumentarium*, has been built in order to compare and classify them according to an accurate conceptual organization. Analysing various models, fundamental functions and features have been identified, isolated and used as a classification basis. The aim of this library is to define generic models or modelling techniques which could be easily used by GENESIS users, whether he or she is a composer or for example a pedagogue who wants to use GENESIS as a support for his or her teaching in Newton's mechanics.

Consequently it is very important to take this into account during the development of our models in order to prefer generic models to ones that use ad-hoc functions.

2.4. The study of self-sustained oscillating structures

Many studies were carried out about physical modelling of self-sustained oscillations of musical instruments with the aim of digital synthesis of real sounds. For example the digital waveguide physical modelling technique was used by Smith, Cook and Scavone to synthesise woodwind [3] [4], bowed string [5] and singing voice sounds [6], or by Karjalainen and Välimäki to model wind instrument bores [7] and vocal tract [8]. The modal synthesis [9] is also a good way to produce this kind of sound.

In the domain of musical acoustics, many researches were undertaken on self-sustained oscillations of musical instruments, which are a good basis for physical modelling in computer music. One can quote inter alia the names of Benade [10] [11] for woodwind instruments or Cremer [12] for bowed strings.

The study presented in this paper, which aims to fill the lack of self-sustained oscillations instrument models in the GENESIS *Instrumentarium*, uses many results obtained by musical acousticians. That is why simple models of bowed strings or woodwinds are presented below, but it is important to notice that our goal is not to model a specific real instrument in the most accurate way but to develop tools that are generic for self-sustained oscillating structures modelling.

3. BOWED STRUCTURES

3.1. A bowed simple vibrating structure

One of the most studied families of instruments is the bowed strings. Thus we will first study the bowing of a vibrating structure in the GENESIS environment. As for all self-sustained oscillations instruments, there is a non-linear element in the instrumental chain of the bowed strings that ensures the production of a high frequency oscillation (vibration of the string) from very low frequency behaviour (movement of the bow). This is the non-linear interaction that takes place between the rosin on hair of the bow and the string. We can see its shape on the graph below:

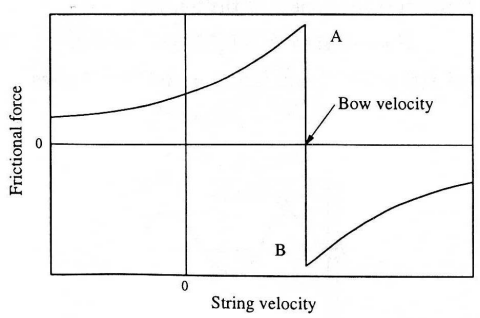


Figure 1: *Frictional force as a function of the string velocity for a bowed string. After Fletcher and Rossing, 1998 [13].*

The LNL module of GENESIS environment let us use this type of interaction since it is possible to draw a F(ΔV) function. Below, you can see the curve that has been chosen:

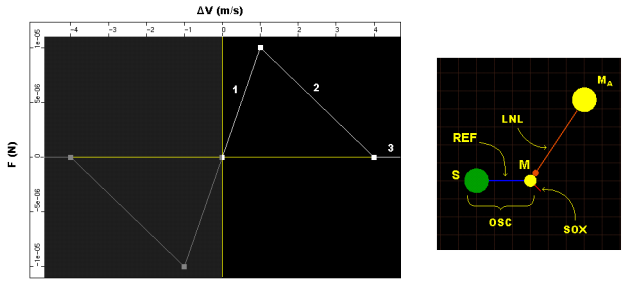


Figure 2: *Left, frictional force as modelled in the LNL window. Right, model of a bowed basic structure.*

This simplified curve that models the interaction between the bow and the string is sufficient to work with, and we will see that it leads to phenomena that are characteristic of real bowed strings behaviours. But the aim is also to use this interaction with other structures than a modelled string; the simplest vibrating structure that we can use is the elementary module CEL. So, we will first work with it in order to illustrate the bowing of oscillating objects. On the figure 2 (right) we can see the representation of the model as it appears on the graphical interface of GENESIS. The MAS module called M_A represents the bow inertia and the structure called OSC which contains a SOL (S), a MAS (M) and a REF link, has got the same behaviour as a CEL module except that it is not optimized. But for a best readability we will use it. So OSC is a damped harmonic oscillator that M_A will bow via the LNL link.

NB: It is important to keep in mind that the representation plan is not a metric space but a topologic one. That is to say, only the links between <MAT> elements will influence the behaviour of our model, not how the <MAT> elements are placed on this plan. Furthermore, the <MAT> modules can move along the axis perpendicular to this plan and only along this axis. That is why GENESIS is called a one-dimension simulation environment. But it is generally not a problem for sound synthesis since oscillations develop themselves mainly on a single axis and it is possible to take into account two or three dimensions effects via LNL links or judicious use of modularity.

We can separate half of the symmetrical friction curve into three parts (noted 1-3 on the figure 2.a)). The first one is called the "sticking zone" and the second one the "sliding zone". For a real bow, the slope of the sticking zone is almost infinite (cf. figure 1) but if we use such a characteristic, the value of the equivalent viscosity Z (i.e. the value of the slope) is almost infinite too. That is why we must use a finite slope unless the algorithm diverges when the difference of velocity is such as the operating point is in the sticking zone of the curve, leading to a sound with more or less white noise (that can get a certain interest). Furthermore, as McIntyre, Schumacher and Woodhouse say in [14] the finite slope of the sticking zone can partially take into account the effects of torsional waves along the string.

Moreover, we must take into account the particularity of our model of interaction. For example, if we start from an oscillator that is at rest and a bow that has got a constant velocity, this velocity must be included between the two boundaries of the sliding zone to obtain a self-sustained oscillation. Indeed, if the velocity is in the third part, no force is applied on the oscillator, and if it is in the first part, no sliding friction can occur and the movement of the oscillator is quickly stabilised in an elongation position that depends of its stiffness (cf. figure 3).

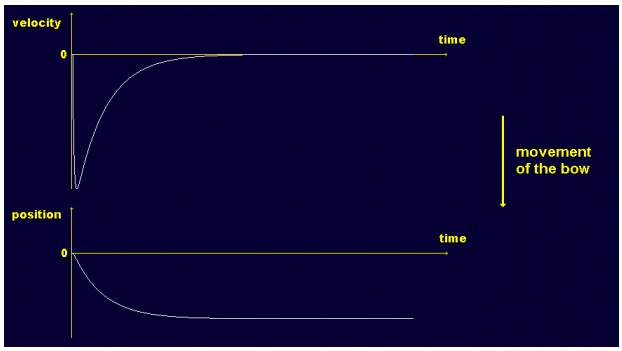


Figure 3: *Velocity and position signals for the bowed oscillator described on figure 2, with a bow velocity in the sticking zone of the LNL characteristic. No oscillation occurs. The behaviour is the one of a damped harmonic oscillator in aperiodic regime (exponential decrease).*

For the bow velocity in the sliding zone of the LNL characteristic, a self-sustained oscillation is obtained as we can see on the figure 4. This fact is due to the negative slope of the curve in the sliding zone. We can see on the velocity signal, for each period, when the operating point passes from the sticking zone to the sliding one (inflexion point, see figure 4). One can note that before this inflexion point, we can see the same behaviour as when the velocity of the bow is in the sticking zone (exponential decrease). After this point, the velocity increases drastically because of the sliding friction; this leads to oscillations.

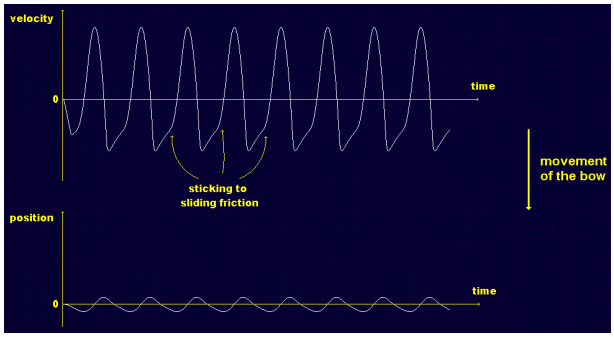


Figure 4: *Velocity and position signals for the bowed oscillator described on figure 2, with a bow velocity in the sliding zone of the LNL characteristic. We can see on the velocity signal the change of behaviour when the system goes from sticking to sliding friction.*

Another parameter that we must precisely adjust is the sliding zone slope, that is the negative damping coefficient value (we note it Z_{neg}). Indeed, if the absolute value of this parameter is lower than the positive damping (Z_{pos}) of the vibrating structure, the self-sustained oscillations regime cannot develop. This can be understood by adding the straight characteristic of the oscillator damping to the one of the LNL link. Indeed, the undamped oscillator will come under the sum of these two characteristics. If the positive damping is higher than the absolute value of the negative one, the sum of the two characteristics will be separated in three parts too, but all with a positive slope. So this situation can be compared to the one where the bow velocity is in the sticking zone and the vibrations of the oscillator quickly decrease. One can compare this behaviour to the minimum bowing force that must be applied on a real string in order to obtain self-sustained oscillations. For low forces, the sliding friction zone has got a very low slope and cannot compensate the damping of the string. A minimum bowing force is thus needed.

Furthermore, if the absolute value of Z_{neg} is higher than Z_{pos} but if these two values are comparable, the transient is very long with a percussive attack at its start. So to quickly obtain a self-sustained oscillations regime, the absolute value of Z_{neg} must be much higher than Z_{pos}.

3.2. Generalisation to other structures

The effects noted for this simple oscillator can be generalised for more complex vibrating structures. For example, we modelled a string by a chain of MAS linked by REF modules. This chain is fixed at both extremities to SOL elements.

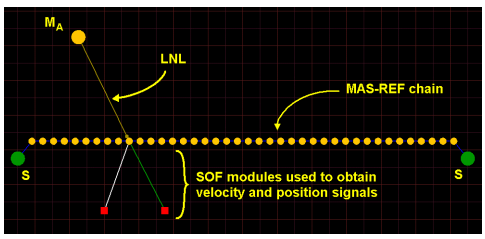


Figure 5: *Model of a bowed string. The two SOF modules are linked to a MAS via RES and FRO modules in order to obtain the velocity and position signals at the point of bowing.*

If we give the correct values to the parameters that we spoke about in the simple oscillator study, the bowing of this structure leads to the well-known Helmholtz motion of the string as we can see on the figure 6.

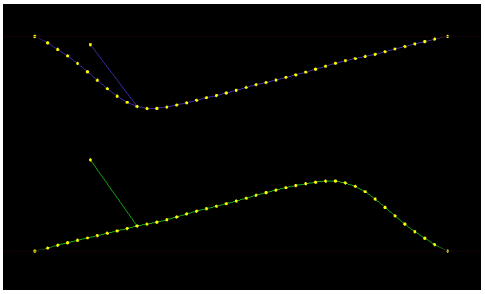


Figure 6: *The Helmholtz motion of the string at two moments which have got a difference in phase of a half period. The bow is moving up at a constant velocity.*

Furthermore, position and velocity signals at different points of our chain are comparable to experimental measures on real bowed strings (cf. figure 7 and 8).

Figure 7: *Velocity and position signals taken at the bowing point for our bowed string model. The bowing point is at a quarter of the string.*

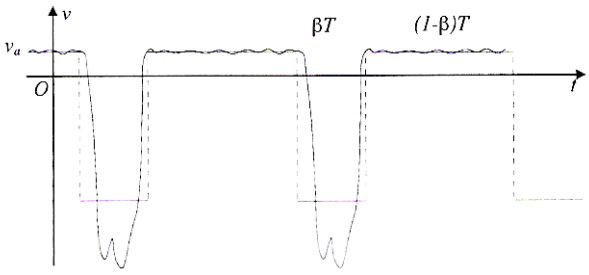


Figure 8: *Velocity of the real string at the bowing point. β is the ratio of the distance between the bowing point and the bridge, upon the length of the string. v_a is the velocity of the bow. After Boutillon, 2000 [15].*

So, the simplified friction characteristic used in our model is sufficient to obtain realistic behaviours and moreover to get plausible bowed string sounds. Note that the real friction force does

not tend to zero when the difference between the bow velocity and the string one is high, whereas it does in our model. The aim is to be able to produce particular gestures like a bow that ends without the bow on the string (in order to be able to produce the sound of the free motion of a string after bowing). Indeed, if we want to cut the link between the vibrating structure and the MAS M_A, we just need to accelerate it until the operating point is always in the third part of the friction characteristic.

So the LNL link described in this part can be used to bow many different structures such as strings, bars, membranes… But as we will see in the next part, this LNL link may be relevant for woodwind instruments modelling too.

4. A PARTICULAR BOWED STRUCTURE

Now, if we take our previously developed string model and link only one of its extremities to a SOL module, the produced sound when we bow the free extremity (using the same LNL as above) sounds like a clarinet. In order to explain this, we can analyse the non-linear characteristic of a woodwind reed (cf. figure 9). It represents the volume flow through the reed as a function of the difference of pressure between the player's mouth and the reed.

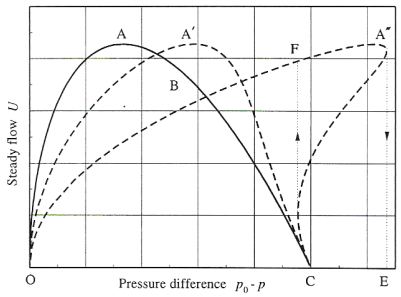


Figure 9: *Characteristic of volume flow as a function of pressure difference for a woodwind single reed (OABC curve) and a woodwind double reed (one of the three curves, according to the reed channel resistance). After Wijnands and Hirschberg, 1995 [16].*

N. B: A remarkable fact is that the friction characteristic of the LNL module developed previously can easily approximate the shape of the curve above, with the help of an analogy that we explain below.

The analogies between mechanical systems and aeroacoustical ones are well known and have been developed in many acoustics books [17]. First of all, the comparison between our LNL characteristic and the curve above suggests that the force applied on and the velocity of the MAS module are respectively the analogue of the volume flow and the pressure inside the reed. But in order to be more precise, let us consider two fluid tanks at different pressures P1 and P2, connected by a channel where a volume flow U of fluid circulates. According to the Euler's equation, we have got in this case:

$$\rho \frac{dv}{dt} = -\frac{dp}{dx} \Rightarrow \Delta P = P1 - P2 = \frac{L\rho}{S}\frac{dU}{dt}, \qquad (2)$$

with L and S respectively the length and the section of the channel, v the speed of the fluid particles and ρ its density. One often calls the factor Lρ/S the acoustic mass. The expression connecting the difference in pressure between the two tanks and the volume flow is similar to the one connecting the difference in speed between two masses connected by a spring:

$$\Delta v = \frac{1}{k}\frac{dF}{dt}, \qquad (3)$$

with Δv the difference in speed between the two masses, K the stiffness coefficient of the spring and F the modulus of the force applied on the two masses. One can then carry out the analogies gathered in the following table:

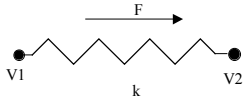
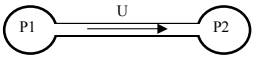
Mechanical system	Aeroacoustical system
	
$\Delta v = \frac{1}{k}\frac{dF}{dt}$	$\Delta P = \frac{L\rho}{S}\frac{dw}{dt}$
F	U
v	P
1/k	$L\rho/S = M_a$

Table 1: *Analogies between mechanical and aeroacoustical systems.*

These analogies let us develop easily woodwind instruments models with mass-spring networks. Indeed, just as our strings are modelled by a succession of masses connected by springs, the body of the wind instruments can be seen as a succession of tanks connected by cylindrical channels.

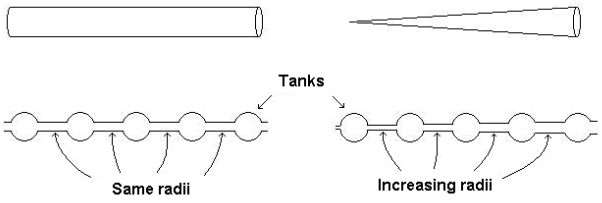


Figure 10: *Simple models of cylindrical and conical bores for wind instruments. In the second case, on the right, the channels have increasing radii in order to model the widening of the bore.*

So, one can translate now this schematized aeroacoustical model into a mass-spring system by means of the developed analogies. On the figure below, we can see the GENESIS model that can be used for woodwind sound synthesis.

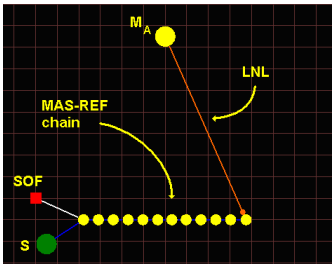


Figure 11: *Woodwind as modelled in GENESIS. The non-linear characteristic used is the same than for the bowed string. The left part of the mass-spring chain is fixed to a SOL module and represents the bell. The other one represents the embouchure.*

So the mass-spring chain is bowed at its free extremity, that is to say, where the v/F ratio is the highest. This is coherent with the behaviour of woodwind instruments for which the P/U ratio is the highest at the reed. On the contrary, a fixed point will represent a hole in the bore. So the SOL at the left extremity represents the hole of the bell. It is possible to model the tone holes too, by adding SOL modules linked to masses along the chain.

As we said above, it sounds like a clarinet for a homogeneous mass-spring chain. This is understandable since the clarinet has got a cylindrical bore. Thus, it might be interesting to try to model other bores, for example a conical one, to obtain oboe-like sounds. The section of the bore of the oboe increases like the square of the distance to the mouth (since its diameter is proportional to the latter). The analogue of the section S is the constant of elasticity K (with a constant factor Lρ). Thus, by giving values, according to a parabolic law, to the K parameters of the consecutive REF modules, it is possible to obtain oboe-like sounds.

On the figure below, it is possible to compare the differences of behaviour between the homogeneous string model (called CLARINET) and the non-homogeneous one (called OBWA).

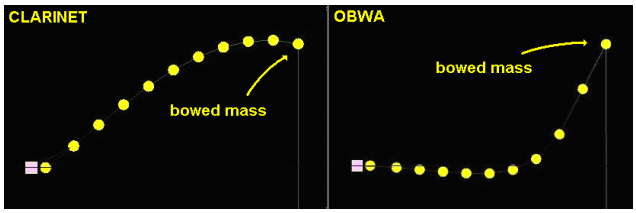


Figure 12: *Aspect of the mass-spring chain at the same phase for the cylindrical bore model (left) and the conical one (right).*

The figure above shows that the behaviours of the two models are not the same. In the first case, the string moves as a whole, the masses of the model being at every moment all on the same side of the rest plan of the string. This shows the prevalence of the fundamental mode on the other ones. On the contrary, for the second model, the mass on which is the excitation is often in opposition of phase with part of the string. This fact is confirmed by spectra of the sounds obtained. These are presented on figure 13 and a comparison is done with experimental data taken in the reference [13]. It is also possible to compare these with the results given in the chapter 21 of the reference [10].

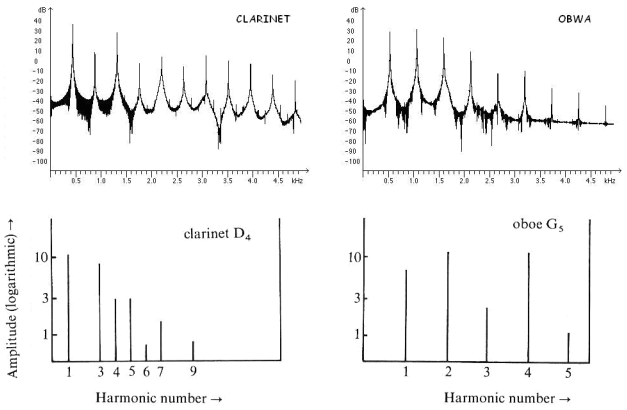


Figure 13: *Spectra of the sounds that we obtained with the CLARINET and OBWA models and comparison with experimental data on real woodwinds. After Fletcher and Rossing 1998 [13].*

So, as for the real instruments, the fundamental prevails for the CLARINET whereas the second harmonic does for the OBWA.

Furthermore, as for a real clarinet, the sound obtained with our CLARINET model has got prevalent odd-numbered harmonics. Even-numbered harmonics are not absent of the spectrum, which has been explained in different references [10] [18].

The analogies developed in this part are very useful since an air column will be simply modelled by the same modules than a string. So it will be very easy to couple structures like strings or membranes with a tube: we only need a <LIA> module. Thus, one can hear for example an oscillating structure vibrating through a duct that has got vocal formants in order to produce vocalizing sounds. This example illustrates the coherence of CORDIS-ANIMA as a general formalism; there is no need to deal with the compatibility of the different models that we develop since the language itself ensures the compatibility.

5. THE BOWING OF MACROSTRUCTURES

This last section deals with features and tools that we can develop in the GENESIS environment by using bowed macrostructures, in order to create events at macrotemporal (compositional and instrumentalist performance) scale. The “macrostructure” term is used to talk about structures that can vibrate at very low frequencies and so that can model the instrumentalist’s gestures.

The underlying idea is that everything that has got inertia is modelled by a MAS module in GENESIS. Consequently, the MAS module, used to model the bow in our models above, can itself be a part of a vibrating macrostructure, which can lead to a complex movement of our bow.

For example, if we consider a string, as in the second part, but with a ratio M/K much higher in order to obtain low frequency modes (~1Hz), and if different MAS modules of this string are used to interact with vibroacoustical structures, it is possible to create a complex play with this macrostructure. On the figure below, we can see such a model, with a “macrostring” that contains plectra, as it has been built in GENESIS.

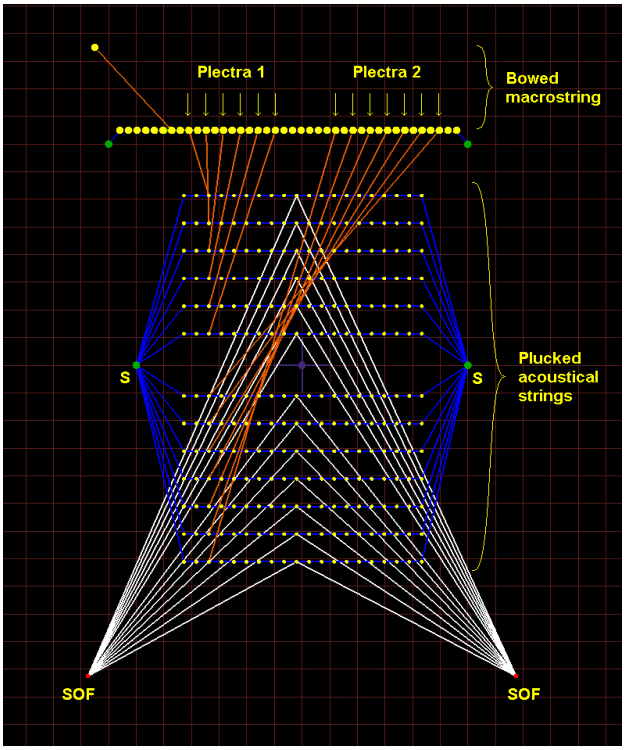


Figure 14: *Model implying a "macrostring" which contains thirteen plectra playing on different acoustical strings.*

The model, as it appears on the figure 14 has been conceived in order to produce a particular play going from low to high acoustical frequencies. Indeed, from top to bottom, the thirteen acoustical strings' fundamental frequency increases. So we have separated these into two groups, each one plucked by a type of plectra (1: low frequency strings, 2: high frequency strings). The first type of plectra corresponds to a LNL module which is calibrated to obtain plucking when the MAS modules of the "macrostring" are at a precise negative altitude "x=-a" (the acoustical strings are in the "x=0" plan), which is the altitude reached by the "macrostring" during its very long transient (cf. figure 15). The second type of plectra is calibrated to pluck when the MAS modules reach the "x=0" plan. On the figure 15 and 16 we can see the advantage of working with two plectra groups. Indeed, the string behaviour is typical of a bowed string transient. But for this system, the transient is very long because of the very low frequency of the string oscillation. So what we have is a movement between two plans. It may be interesting to use some plectra for the moment when the string is in one plan and other plectra when it is on the other plan.

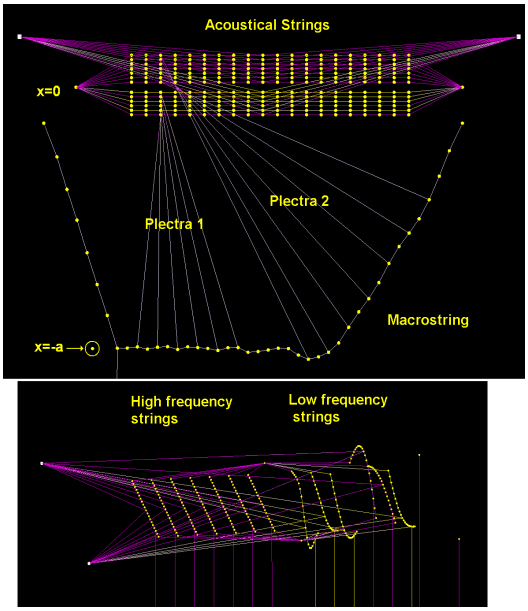


Figure 15: *Two different viewpoints of the simulation of the model shown on the figure 13 at 1,25 second. First phase of the period of the "macrostring" movement. This one goes down until it reaches an altitude located by the MAS circled (top picture). The six plectra on left are calibrated to pluck the low frequency strings at this altitude. So we can see on the bottom picture that these six strings oscillate. On this picture, the vertical scale is much lower than for the top picture.*

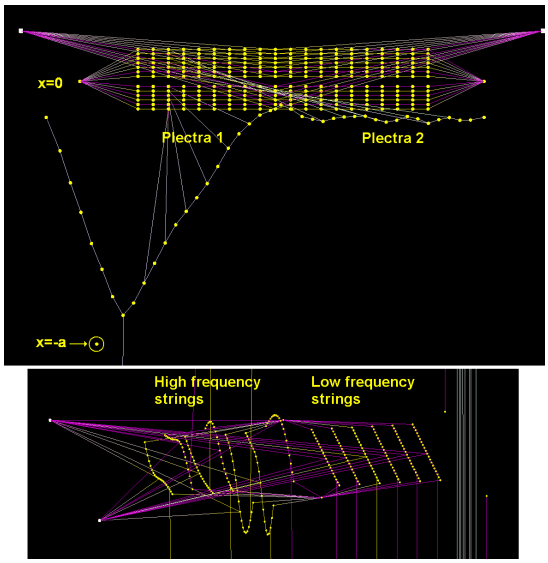


Figure 16: *Two different viewpoints of the simulation of the model shown on the figure 13 at 2,5 seconds. Second phase of the period of the "macrostring" movement. This one goes up until it reaches the "x=0" plan. So now the seven plectra of the second group pluck the high frequency strings as we can verify it on the bottom picture.*

So the "instrumental play" has got a repeated cycle that is divided into two phases: the first when the "macrostring" is at its negative altitude (figure 15) and the low frequency strings are plucked, the second when it is at the zero altitude and the other strings are plucked.

This produces a periodic alternation between complex series of low and high-pitched notes. Furthermore, these musical events evolve in time since the behaviour of the "macrostring" described above is the transient one. Progressively, higher amplitude oscillations take place, and it results in less plucks (but more disorganised). This gives the impression to pass from a vigorous part with lots of musical events to calm and quietness.

Finally, we can say that the bowed "macrostring" produces an "instrumental play" that is not precisely predictable but, so far, not unpredictable either. Its periodic oscillation leads to a pulsation. Moreover the precise analysis of the model's behaviour can give information on how to use it, to privilege a precise note for example. Furthermore it has got very rich possibilities. For instance, it is possible to change the period of the instrumental play by changing the K/M ratio of the "macrostring" or to increase or decrease its transient by influencing the bow's friction characteristic. It is possible to change the acoustical strings damping in order to get more or less resonant sounds, or to bow these ones instead of plucking them…

Finally, one must study in details these sorts of models because in one hand they have got rich possibilities but in the other hand one must wonder: what are the minimum characteristics required to get a relevant model for expressive musical architectures production? There is no doubt that the research on this point with GENESIS is at its infant. But it will be certainly fruitful to carry out researches in this way.

6. CONCLUSION

The self-sustained oscillating structures category is a very useful family of models that is relevant for studies upon both timbre and composition in GENESIS and thus that must be developed and inserted into its *Instrumentarium*. We saw that, by means of analogies, real musical instruments of different natures can be simply modelled by almost the same bowed structure. Moreover, since the same elementary modules are used for the building of the structures, the compatibility of all the models is thus ensured. So it is very easy to couple all our different vibroacoustical structures in order to produce more complex and interesting timbres. The studies will now be carried out on other structures too. For example, structures with lots of modes can produce interesting evolving sounds when bowed repeatedly.

As for the composition in GENESIS, bowed macrostructures offer many possibilities but need to be deeply analysed in order to be used in precise ways. For this, a time analysis of position or velocity signals appears to be more appropriate than a frequency one. In any case, a good comprehension of their behaviours is necessary in order to be able to insert this kind of tools in a musical piece with GENESIS.

7. REFERENCES

[1] N. Castagné and C. Cadoz, "GENESIS: A Friendly Musician-Oriented Environment for Mass-Interaction Physical Modeling". *Proc. of the 2002 Internat. Comp. Mus. Conf.* San Francisco, International Computer Music Association, 2002.

[2] C. Cadoz, A. Luciani and J. L. Florens, "CORDIS-ANIMA: A Modeling and Simulation System for Sound and Image Synthesis – the General Formalism". *Computer Music Journal*, vol. 17, no. 4, pp. 19-29, 1993.

[3] G. Scavone, "Digital waveguide modelling of the non-linear excitation of single-reed woodwind instruments". *Proc. of the 1995 Internat. Computer Music Conf.* pp. 521-524, International Computer Music Association, 1995.

[4] G. Scavone and J. O. Smith, "Digital waveguide modelling of woodwind tone holes". *Proc. of the 1997 Internat. Computer Music Conf.* Greece, International Computer Music Association, 1997.

[5] J. O. Smith, "Efficient synthesis of stringed musical instruments". *Proc. of the 1993 Internat. Comp. Mus. Conf.* Tokyo, pp.64-71, International Computer Music Association, 1993.

[6] P. R. Cook, "SPASM: A Real-Time Vocal Tract Physical Model Editor/Controller and Singer: the Companion Software Synthesis System". *Computer Music Journal*, vol. 17(1), pp. 30-44, 1992.

[7] V. Välimäki and M. Karjalainen, "Digital waveguide modelling of wind instrument bores constructed of truncated cones". *Proc. of the 1994 Internat. Comp. Mus. Conf.* Arhus, pp. 423-430, International Computer Music Association, 1994.

[8] V. Välimäki and M. Karjalainen, "Improving the Kelly-Lochbaum vocal tract model using conical tube sections and fractional delay filtering techniques". *Proc. of the 1994 Internat. Conf. on Spoken Language Processing.* Yokohama, pp. 615-618, IEEE Press, 1994.

[9] J.-M. Adrien, "The missing link: Modal synthesis". *Representations of Musical Signals* (G. De Poli, A. Picialli and C. Roads), pp. 269-297, Cambridge, MA: MIT Press, 1991.

[10] A.H. Benade, *Fundamental of Musical Acoustics*. Oxford U. P., New York, 1976.

[11] A.H. Benade, "On woodwind instrument bores". *J. Acout. Soc. Am.* vol. 31, pp. 137-146, 1959.

[12] L. Cremer, *The Physics of the Violin.* Translated by J. S. Allen, MIT Press, Cambridge, Massachusetts, 1984.

[13] N.H. Fletcher and T.D. Rossing, *The Physics of Musical Instruments*, 2nd Edition, Springer, 2005.

[14] M. E. McIntyre, R. T. Schumacher and J. Woodhouse, "On the oscillation of musical instruments". *J. Acout. Soc. Am.* vol. 74, pp. 1325-1345, 1983.

[15] X. Boutillon, "Corde frottée sur un violon : dynamique, mouvements, standards et instabilités". *Mécanique et Industrie 1.* pp. 609-619, 2000.

[16] A. P. J. Wijnands and A. Hirschberg, "Effect of a pipe neck downstream of a double reed". *Proc. Internat. Symp. Mus. Acoust.* (Dourdan, France), pp.148-151, IRCAM, Paris, 1995.

[17] M. Rossi, *Electroacoustique.* Traité d'électricité, vol. 21, PPUR, 1993.

[18] A.H. Benade and S.N. Kouzoupis, "The clarinet spectrum: Theory and experiment". *J. Acout. Soc. Am.* vol. 83, pp. 292-304, 1959.

FILTERING WITHIN THE FRAMEWORK OF MASS-INTERACTION PHYSICAL MODELING AND OF HAPTIC GESTURAL INTERACTION

Alexandros Kontogeorgakopoulos *Claude Cadoz*

ICA laboratory
INPG, Grenoble, France
alexandros.kontogeorgakopoulos@imag.fr

ACROE, ICA laboratory
INPG, Grenoble, France
claude.cadoz@imag.f

ABSTRACT

A variety of filters have been designed, synthesized and used in the history of electronic and computer music. All the approaches aimed to provide filters fulfilling several specifications such as frequency response, phase response, transient state characteristics like rise time and overshoot, realizably conditions concerning the technology used for the implementation and even economical considerations. One of the most important aspects concerning the filters dedicated to musical applications is the control structure they provide to the musician, who is in charge for the integration of the filtering operation in the compositional process and performance. Designing filters using the mass interaction scheme embedded in the CORDIS-ANIMA formalism (used for sound synthesis and composition by physical modelling) offers a different methodology in the control which is coherent with the philosophy of musical composition by 'physical thinking'. This article introduces a technique to design filters using the CORDIS-ANIMA simulation language.

1. INTRODUCTION

Filters play a crucial role in the history of sound transformation. They have been in use since the very early days of electronic music. The first systematic design approaches date back to first decades of the previous century. Many electronic instruments of the 1930s including the Trautonium, used analogue filters [1]. Filters were standard components in electronic music facilities such as the West German Radio (WDR) studio in which K. Stockhausen, G. M. Koening and other composers worked in the 1950s and 1960s.

In the field of computer music, digital filters first appeared in sound synthesis languages such as Music IV after 1963. Nowadays, among several common used filters are the digital resonator which are special two-pole bandpass filters [2], the state variable filter [3], the simulations of the Moog four pole filter like the one proposed by Huovilainen [4] and the parametric filter structures like the Regalia or the Zolzer filters[5].

A filter structure is expected to give direct or indirect aspects to the perceptual parameters like the center/cutoff frequency, the bandwidth and the gain. This is accomplished by controlling the transfer functions coefficients describing the designed filter. In this scenario the filter is conceived as a mathematical operation that transforms the audio signal. This functional point of view disallows the "Physical Instrumental Interaction" with the system which performs the filtering operation and guides to the general question of mapping between the control signals and the available input parameters of the system – in this case the transfer function coefficients.

What we call " Physical Instrumental Interaction " [6] is here crucial:
It is indeed the type of physical interaction which we establish with a real instrument. In this interaction, the "ergotic function" [7], [8], [9] which is what allows in a direct way to act on the physical instrument and to feel it by the haptic perception, plays an essential role. This lets to perform the gesture in an expressive way and then to produce and even transform expressively sounds. In digital sound synthesis or transformation, the ergotic function can be supported by specific force-feedback gestural transducers [10],[11]."Physical thinking" and "Physical Instrumental Interaction" are very closely associated.

So we can envisage that the filtering process is performed by a simulated physical mechanical system and not by an abstract signal processing algorithm. In this case we are able to establish by involving a suitable ergotic interface, a physical interaction between the musician and the filter which has now a virtual material substance. In this type of control there is no mapping between gesture and sound since no representation is involved in this situation, but only physical processes. The CORDIS-ANIMA simulation system [12], which in fact is a formalism intended for simulating the instrumental relationship, permits to synthesize and control filters using this physical modeling approach.

Therefore the purpose of this study is to synthesize audio filters using CORDIS-ANIMA networks. For the design part of the filter other well known methods were adopted like the pole/zero placements and approximation techniques for all pole filters like the Butterworth approximation [13]. In this essay we designed the filter by putting in parallel a certain number of second order sections: the well known simple two-pole filters. Evidently the actual intention of this research is not to propose a new implementation of filters but to give to filters the character, the nature and "charm" of a physical tangible object that can be manipulated and controlled by physical gestures. These CORDIS-ANIMA filter models are transferred to the GENESIS environment mainly dedicated to musical composition by physical modeling [14] and eventually will be used in combination with force-feedback gestural interfaces for real time musical applications.

2. CORDIS-ANIMA AND GENESIS

CORDIS-ANIMA is a real-time mass-interaction physical modeling and simulation system. This highly modular language was

used during this study to simulate physical models that play the role of digital audio filters. CORDIS-ANIMA allows designing and simulating virtual objects that are composed of two types of elements, called modules:

- <MAT> modules represent punctual material elements. The most used is the MAS module, which simulates an ideal inertia. The <MAT> modules are elementary subsystems and can be characterized in terms of their input/output relationships.

- <LIA> modules represent physical interactions between pairs of <MAT> modules. Available interactions are based on linear or nonlinear elasticity and friction. The <LIA> modules are elementary subsystems and can be characterized in terms of their input/output relationships.

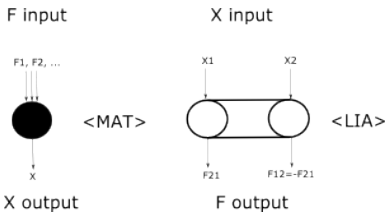


Figure 1 : *<MAT> and <LIA> modules with their inputs and outputs*

Thus, CORDIS-ANIMA models are networks of interconnected <MAT> and <LIA> modules.

Position and force are the two fundamental variables upon which CORDIS-ANIMA modules operate. At each sample a <LIA> computes two opposite forces according to the relative distance and/or velocity of the two <MAT> it links while a <MAT> computes its position according to the forces it receives from the <LIA> modules it is linked with. It should be noticed that some <MAT> modules are fixed points, so received forces have no effect on them. The algorithms can be found on [15]. In figure 1 the <MAT> and <LIA> elements are depicted with their inputs and outputs.

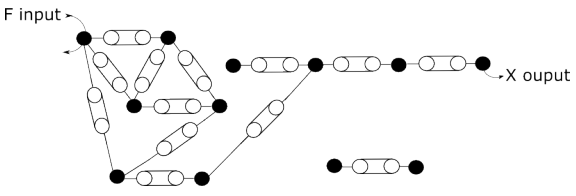


Figure 2 : *A CORDIS-ANIMA network*

GENESIS [16] is a graphical environment for musical creation based on CORDIS-ANIMA. The user builds CORDIS-ANIMA networks at an elementary level, since models are created by direct graphical manipulation and connection of individual modules on a virtual workbench. A number of higher-level tools are available for editing multiple parameters at the same time, generating large structures, visualizing models during simulation, etc. GENESIS implements ten types of modules. While CORDIS-ANIMA does not specify the dimensionality of the modules, GENESIS' simulation space is one-dimensional. <MAT> modules can only move in the direction that is perpendicular to the workbench, and distances and velocities are computed along this axis. For convenience, graphical manipulations take place in the 2D-space of the workbench, but the position of the modules on this plane have absolutely no consequence on the simulation: the workbench representation is only topological.

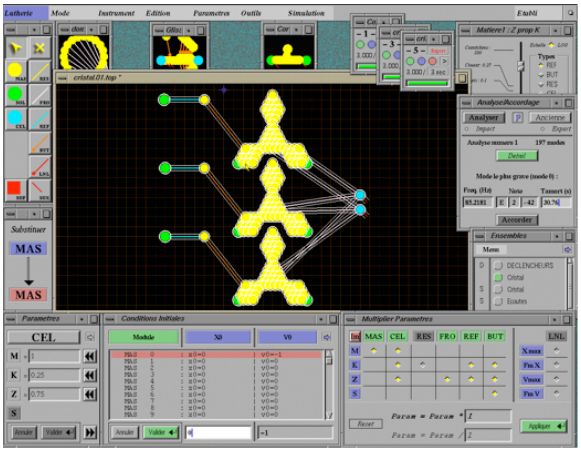


Figure 3 : *A screenshot from the GENESIS software*

The normal set of GENESIS' building blocks is composed of:

- Linear modules: ideal mass (MAS), fixed point (SOL), second-order damped oscillator (CEL), elasticity (RES), friction (FRO), elasticity and friction combined (REF);
- Nonlinear interactions: the BUT and the LNL;
- Output modules: the SOX and the SOF, which respectively record a position and a force signal.

The BUT module simulates a conditional viscoelastic interaction. Installed from one MAS (M1) to a second one (M2), when the difference between the positions of M_1 and M_2 is smaller than a given threshold S, the BUT simulates the effect of a null-length damped spring between M_1 and M_2; otherwise, the two modules are not linked.

The LNL module is a user-defined nonlinear viscoelastic interaction. The user chooses the points defining two curves and may interpolate them using linear interpolation, splines, hyperbolic interpolation. The first curve (LNLK) gives the force to be applied to the modules according to the difference of their positions (nonlinear elasticity). The second curve (LNLZ) gives the force according to the difference of their velocities (nonlinear friction).

All <MAT> modules have an initial position (X0). Mobile <MAT> modules also have an inertia parameter (M) and an initial velocity (V0). <LIA> modules have elasticity (K=k/Fs^2 – k measured in S.I, Fs the sampling rate) and/or friction (Z=z/Fs, z measured in S.I) parameters.

During this study, we used a particular version of GENESIS that includes two extra modules, ENX and ENF. These are input modules that read an input file and translate its data into a time-changing position (ENX) or force (ENF). The input file represents a 1D temporal signal, sampled at 44100 Hz. It may derive from the measurement of a real gesture, the recorded movement of a <MAT> module in a previous simulation, or from an audio file. Consequently, input modules can be used to input any audio signal into GENESIS' models.

ENX is a massless <MAT> module whose position corresponds at each moment to the last sample read in the input file. ENF is a <LIA> module that connects to a single <MAT>, to which it sends a force proportional to the input file data.

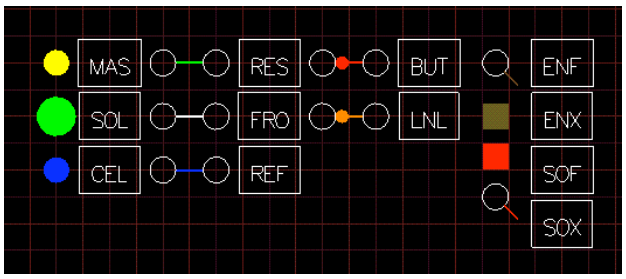


Figure 4 : *GENESIS basic modules*

3. GENERAL ASPECTS OF DESIGNING AND IMPLEMENTING DIGITAL FILTERS WITH CORDIS-ANIMA

The basic idea behind any digital audio effect and in our case the filters using physical modeling techniques are the forced oscillations. Sometime, it happens that a system is put into vibration because it is linked with another oscillating system which is called the driving system. The important feature on forced oscillations is that the driven system does not feed back any appreciable amount of energy to the driving system so the perturbation effects are negligible. The input sound for the digital audio effects takes the role of the driving system.

In the simulated world of CORDIS-ANIMA there are two ways to connect two mechanical systems. It is possible to consider as the output of the first system and as the input of the second system either the force or the position, according to the fact that the first is of <LIA> category or of <MAT> category, even though the variables in the Newtonian mechanics are duals and are not separable. On the other hand the computer simulation and the real time control force the separation of these variables. In general all physical communications (which are intrinsically non-oriented) are presented by two-way communication carried out by divisible input/output pairs (this is a constraint inherited by the information theory and the capabilities of the technology) [12]. A special case is envisaged for the force oscillations where the driving force or driving position is not presented by divisible input/output pairs but with a single input communication channel. Therefore we are able to apply directly a force to a <MAT> module or a position to a <LIA> module by an input file acting as the input sound (ENF and ENX modules in GENESIS – used only as external input). In a similar situation, whereas the physical model acts as the driving system, we can deliver a position using a <LIA> where it returns a zero force to the model or we can deliver a force using a <MAT> where it returns a zero position to the model (SOX and SOF modules in GENESIS – used only as external output). In this case the position or the force are considered as the output signal and it is recorded in sound files.

In reality we always have feedback links between interacting mechanical systems. However it is possible to reach situations of forced vibrations when we link mechanical systems where we approximately ignore the feedback of energy either because the linkage is very weak or else because the driving one has so much reserve energy that the amount of feedback is comparatively negligible [17] . So in CORDIS-ANIMA models, we can control the feedback interconnection following this principle and approximately pass from feedback interconnections to feed-forward ones. We are able to do this either by changing the impedance of the systems (the one with the considerably higher impedance drives the other) or by using a weak link.

4. CAUER REALIZATIONS FOR LC ELECTRICAL CIRCUITS

The Cauer synthesis procedure of passive electrical networks concerns the implementation of a specified immittance function by a particular form of ladder electrical networks. It is one among several other methods used for the synthesis of driving point immittances [18]. Immittance is a general term used to include both impedance and admittance. In many cases the required immittance is realized using only LC circuit elements: inductors L with $Z_L(s) = Ls,\ Y_L(s) = \frac{1}{Ls}$ and capacitors C with $Z_C(s) = \frac{1}{Cs},\ Y_C(s) = Cs$.

The necessary conditions that must by satisfied by a rational function that is realizable as the LC driving-point immittance may be synopsized [19]:

- The poles are simple and on the $j\omega$ axis
- The zeros are simple on the $j\omega$ axis
- The poles and zeros alternate
- There is a pole or a zero at the origin
- There is a pole or a zero at infinity
- The residues of the poles are real and positive
- The functions are reactance functions whose value along the $j\omega$ axis is purely imaginary, i.e. $Z_{LC}(j\omega) = jX(\omega),\ Y_{LC}(j\omega) = jB(\omega)$
- $dX(\omega)/d\omega$ and $dY(\omega)/d\omega = jB(\omega)$ are always positive
- The functions are odd rational functions which mean that if the numerator is an even polynomial then the denominator is an odd polynomial and vice-versa.

The general form of a realizable LC driving-point immittance I is

$$I_{LC}(s) = \frac{k_0}{s} + k_\infty s + \sum_i \left[\frac{c_i}{s - j\omega} + \frac{c_i^*}{s + j\omega}\right] \quad c_i = c_i^* \Rightarrow$$

$$I_{LC}(s) = \frac{k_0}{s} + k_\infty s + \sum_i \frac{2c_i s}{s^2 + \omega^2}$$

where k_0, k_∞, c_i are the residues of the poles at the origin, at the infinity and on the $j\omega$ axis, respectively.

The ladder network has a specific topology with alternating series and shunt branches as shown in figure 5. This singularity allows the driving-point immittance to be expressed in the following form [20]:

$$Z = Z_1 + \cfrac{1}{Y_1 + \cfrac{1}{Z_2 + \cfrac{1}{Y_2 + \ldots}}}, \quad Y = Y_1 + \cfrac{1}{Z_1 + \cfrac{1}{Y_2 + \cfrac{1}{Z_2 + \ldots}}}$$

The heart of this method follows from considering an LC driving-point immittance which consequently has poles at infinity, the origin and complex-conjugate poles on the $j\omega$ axis. Removing any poles of this function results in a function which is as well LC realizable.

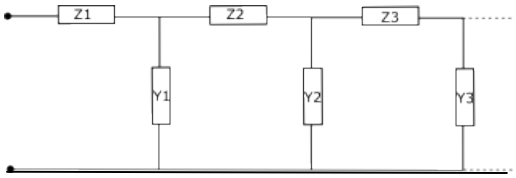


Figure 5 : *Ladder network*

Each division can be carried out starting either with the highest or with the lowest power of s. When each division starts with the highest powers, the procedure is known as *Cauer I* method. On the other case the procedure is known as the *Cauer II* method. If the order of the numerator is greater than the order of the denominator the *Cauer I* method is used which always leads to a ladder with series inductors and shunt capacitors. In a similar way if the order of the numerator is smaller than the order of the denominator the *Cauer II* method is used which always leads to a ladder with series capacitors and shunt inductors

For example the admittance function $Y(s) = \frac{s^4+4s^2+3}{s^3+2s}$ can be expanded in the form $Y(s) = s + \cfrac{1}{s/2 + \cfrac{1}{4s + \cfrac{1}{s/6}}}$

This corresponds to a realizable circuit with Y_{C1}=1s, Z_{L1}=1/2s, Y_{C2}=4s, Z_{L2}=1/6s.

5. SYNTHESIS BASED ON CAUER TECHNIQUE

Every CORDIS-ANIMA model has an analogue Kirchhoff network in the continuous time domain. This analogy, permits in a certain number of cases to apply the Cauer method in the CORDIS-ANIMA simulation system. A detailed analysis concerning the Kirchhoff/CORDIS-ANIMA analogy is out of the scope of this article. In figure 6, the analogue of a LC ladder network is presented by the CORDIS-ANIMA network topological diagram. For this case, the analogy displayed in the table 1 was used. Z, K, M are the variables used in GENESIS.

force F ⇔ potential U	$m = L \Rightarrow M = L$
velocity V ⇔ current I	$k = \frac{1}{C} \Rightarrow K = \frac{1}{CF_s^2}$
position X ⇔ charge Q	$z = R \Rightarrow Z = \frac{R}{F_s}$

Table 1 : *CORDIS-ANIMA/Kirchhoff analogy*

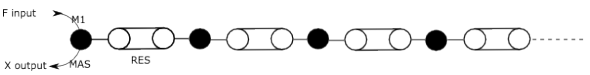


Figure 6 : *LC Ladder network in CORDIS-ANIMA*

The double discretization scheme adopted by CORDIS-ANIMA unfortunately does not permit the use of a direct transformation method from the s-domain to the z-domain. Nevertheless we may use an indirect method using the modal decomposition. The desired filter function is designed directly in the discrete time simulation space of CORDIS-ANIMA using a bank of second order parallel resonators. Each resonator is tuned to a certain resonating frequency F_i and Q-factor Q_i. We remark that in this research the peak of the resonators is approximated by their natural frequencies. From these filter perceptual characteristics we compute the physical characteristics i.e. the mass M, the elasticity K and the friction Z. Approximately M affects the amplitude of the filter, M/K the resonant frequency and Z the Bandwidth or the Q parameter. The adopted Cauer synthesis procedure is for LC networks and so non-dumped structures were treated and synthesized. In this case $Z = 0 \Rightarrow Q = \infty$. Using the M and K physical parameters or the m and k in the S.I system we form the second order parallel resonators on the continuous time. Having computed the impedance function we are ready to apply the Cauer technique. A detailed description of the algorithm is depicted below.

Algorithm

$$Y_{CA_i}^{sos}(z) = \frac{X_i(z)}{F_i(z)} = \frac{1/(M_i F_s^2) z^{-1}}{1 + (K_i/M_i - 2)z^{-1} + z^{-2}} \rightarrow$$

1. $K_i / M_i = 2 - 2\cos\left(\frac{F_{CAi} 2\pi}{F_s}\right)$

   where F_s *= sampling rate,* F_{CAi} *desired frequency in Hz*

2. $Y_i^{sos}(s) = \frac{Q_i(s)}{V_i(s)} = \frac{1}{L_i s^2 + 1/C_i} \rightarrow Y_{all}(s) = \sum_i Y_i^{sos}$

   where $L_i = M_i$, $C_i = 1/(K_i F_s^2)$

3. $Z_{Cauer}(s) = \frac{V(s)}{I(s)} = \frac{V(s)}{Q(s)s} = \frac{1}{Y_{all}(s)s} \Rightarrow$

   $$Z_{Cauer} = Z_1 + \cfrac{1}{Y_1 + \cfrac{1}{Z_2 + \cfrac{1}{Y_2 + \ldots}}}$$

4. $\left.\begin{matrix} Z_i(s) = L_i s \\ Y_i(s) = C_i s \end{matrix}\right\} \begin{matrix} M_i = Z_i(s)/s \\ K_i = s/(Y_i(s) F_s^2) \end{matrix}$

It is possible to reach a more approximative solution if we start the algorithm from the step 2: We can tune directly the second order sections in the continuous time domain:

$$F_i = \frac{1}{2\pi}\sqrt{k_i/m_i} \rightarrow k_i/m_i = (F_i 2\pi)^2$$

where $L_i = m_i$, $C_i = 1/k_i$

More details concerning the second order sections for the continuous time and the discrete time case can be found in [3], [13], [21].

In the following graph (figure 7) the deviation in the resonating frequency is depicted between the 2-pole filter in the continuous time domain and after the discretization scheme used in CORDIS-ANIMA. We observe that for frequencies smaller than 1000 Hz the difference is negligible-less than 1 Hz. So when we design filters in the region 0-1000Hz we can design them directly in the continuous time domain.

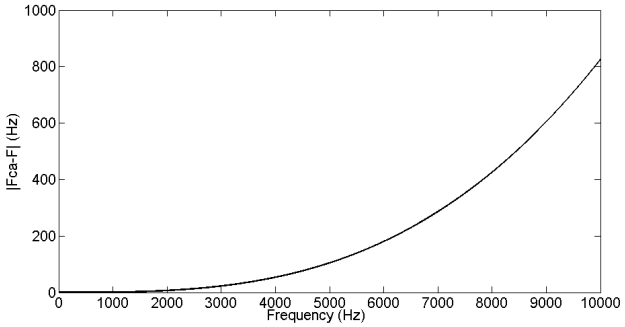


Figure 7 : *Deviation in the resonating frequency*

For a Cauer expansion to correspond to a realizable ladder network, all the coefficients of the expansion must be positive. The CORDIS-ANIMA simulation system on the other hand, permits to use negative coefficients.

6. RESULTS-A TANGIBLE FILTERING PARADIGM

Several filters were synthesized using the previous algorithm. All the scripts were written in matlab® and the results were transferred to GENESIS. We will present a simple example of a filter with resonating frequencies 200Hz, 240Hz, 450Hz and 530Hz and $Q = \infty$ which means that the filter will starts ringing in its resonating frequencies. The filter is designed with method described in the previews chapter. It has an admittance function given by the following formula:

$$Y_{CA}(z) = \frac{4 - 23.9646z^{-1} + 59.8586z^{-2} - 79.7880z^{-3} + \ldots}{1 - 7.9882z^{-1} + 27.9293z^{-2} - 55.8233z^{-3} + \ldots}$$

$$\ldots \frac{59.8586z^{-4} - 23.9646z^{-5} + 4z^{6}}{69.7645z^{-4} - 55.8233z^{5} - 27.9293z^{-6} - 7.9882z^{-7} + z^{-8}} \rightarrow$$

$$Y(s) = \frac{4s^{6} + 6.879e07s^{4} + 3.313e14s^{2} + 4.098e20}{s^{8} + 2.293e07s^{6} + 1.657e14s^{4} 4.098e20s^{2} + 3.18e26}$$

We verified that the algorithm is very accurate. The final physical model has the same admittance function as with that one we started from in the design phase of the filter. In figure 8 is illustrated this admittance function.

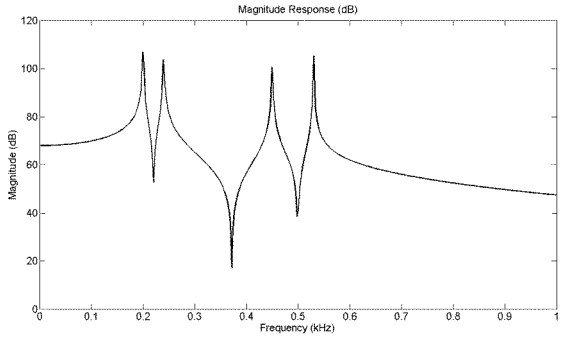


Figure 8: *Graph of the admittance function of the CORDIS-ANIMA realization of the filter*

The block diagram in figure 9 depicts a parallel form realization of the filter which is widely used in the digital signal processing domain. The control is based on the direct access to the parameters of the structure: the multipliers of the feedforward and the feedback paths. Every possible method can be used to translate the user actions into these parameters value.

The block diagram in figure 10 presents a CORDIS-ANIMA realization of the filter. This structure offers directly another type of control based on the "Physical Instrumental Interaction". We don't affect the parameters of the model -even though it is possible and previewed within the CORDIS-ANIMA system- but we apply forces to the <MAT> elements of the model using <LIA> elements. In this example we can interact physically with the masses M1, M2, M3 and M4.

It is straightforward that this type of control is totally physical and energetic coherent. Since physical models enable an intuitive representation of the action we perform with real objects we can imagine several physical gestures to play with our filter: dumping, pulling, pushing, e.t.c. This is still true for non real-time simulations and without the use of force feedback gestural interfaces but by designing models that simulate the physical gesture. The deferred-time simulation permits to design accurate and valid models of the control gesture with a precision that is not possible in the real-time situations. Figure 11, which is a snapshot taken for the GENESIS software, illustrate the physical model for the filtering operation described earlier and a physical model for the physical control of the filter. For the control we use a periodical gesture that dumps the movement of the string used as a filter when it reaches it.

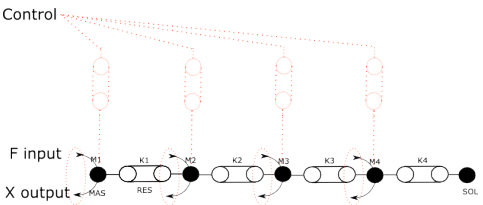


Figure 10: *A CORDIS-ANIMA filter realization*

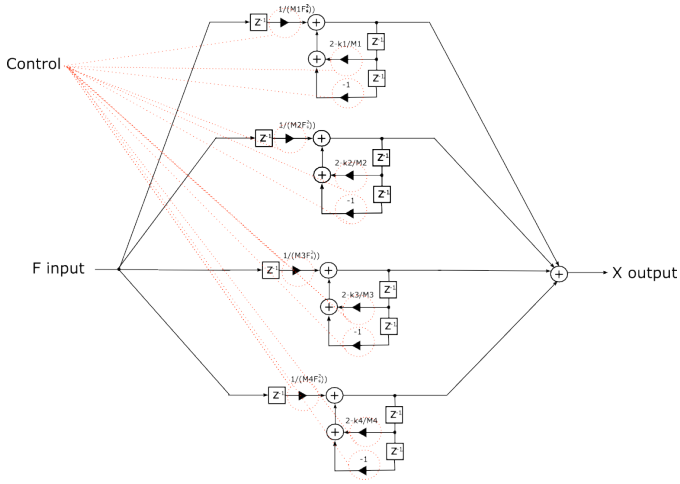


Figure 9: *A parallel form filter realization*

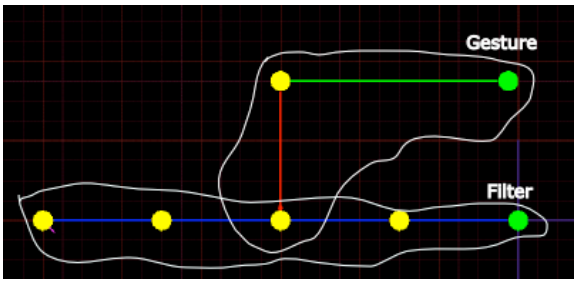


Figure 11: *GENESIS example of a virtually mechanical filter controlled by another physical structure*

7. CONCLUSION AND FUTURE WORKS

A method to synthesize filters using the mass interaction scheme was presented. Adopting the Cauer technique from the domain of the electrical networks, we designed physical models for filtering purposes. This permitted to benefit some of the advantages of the physical modeling in the audio signal filtering domain like the control based on physical interactions.

Even if the Cauer synthesis procedure goes further from the driving point immittance functions and may reach the implementation of a specified transfer function, in this study it was not crucial to reach this point. The main intention was more to tune up physical structures – strings in our case to a pre-given set of frequencies than to realize and implement a given transfer function. Several other methodologies were also examined. Amongst them, optimization algorithms were used i.e. the Newton method, which provides less precision but is much more general and can be applied for all kind of structures like pyramids, spirals, membranes [22]. These results could be the subject of another article. It is clear that the methodology using the optimization algorithms can be viewed as special case of the inverse problem: the values of some model parameters i.e. M, K and Z must be obtained by the observed/desired data.

The synthesis technique presented in this study may easily be transferred to other widely used physical modeling approaches like the digital waveguides [23]. In our case all the needed scripts were written in matlab® and the results were transferred in GENESIS environment for further physical manipulation, control and compositional research based on "physical thinking".

The applications of physical modeling for sound synthesis are numerous. However the power of physical modeling for sound processing has not been explored yet. Filtering is the basic signal manipulation mechanism so it is straight forward why this research concerned audio filters and physical models.

This article is a part of a wider research focused on the design of physical models that would transform and process sounds using the CORDIS-ANIMA formalism. The objective is to offer to the transformation procedure an instrumental "character" with the purpose of hopefully getting more "warm" or "live" audio effects, and setting up a relation between the system and the musician of the type instrument/instrumentalist.

8. REFERENCES

[1] Harald Bode, " History of Electronic Sound Modification", *Journal of the Audio Engineering Society*, 32(10), 1984

[2] Udo Zolzer, *Digital Audio Effects*, John Wiley & Sons Ltd, 2002

[3] J. O. Smith, *Introduction to Digital Filters, September 2005 Draft,* http://ccrma.stanford. edu/~jos/filters05

[4] A. Huovilainen, "Non-linear digital implementation of the Moog ladder filter", in *Proceedings of the 2004 Digital Audio Effects Conference* pp. 61-64, 2004

[5] Udo Zolzer, *Digital Audio Signal Processing,* John Wiley & Sons, 1997

[6] C. Cadoz, "Instrumental Gesture and Musical Composition", in *Proceedings of the 1988 International Computer Music Conference*, pp. 1-12, International Computer Music Association, San Francisco, 1988

[7] J. Boissy, *Cahier des termes nouveaux*, Institut National de la Langue Française, Conseil International de la Langue Française (CILF) and CNRS Editions, pp. 52, 1992

[8] C. Cadoz, "Le geste, canal de communication instrumental", *Techniques et Sciences Informatiques* Vol 13 - n01 pp. 31-64,1994

[9] C. Cadoz, M. M. Wanderley, "Gesture-Music", in *Trends in Gestural Control of Music*, M. M. Wanderley and M. Battier, eds, ©2000, Ircam – Centre Pompidou, pp. 71-94 , 2000

[10] C. Cadoz, A. Luciani and J.-L. Florens, "Responsive Input Devices and Sound Synthesis by Simulation of Instrumental Mechanisms : The CORDIS System", *Computer Music Journal* 8(3), 1984 – reprint in *The Music Machine* – edited by Curtis Roads, The MIT Press, Cambridge, Massachusetts, London, England - pp. 495-508, 1988

[11] A. Luciani, C. Cadoz, J.-L. Florens, "The CRM device : a force feedback gestural transducer to real-time computer animation", *Displays: technology and applications*, 1994/07, Butterworth - Heinemann, vol. 15 number 3 - pp. 149-155, 1994.

[12] C. Cadoz, A. Lucian and J.L Florens, "CORDIS-ANIMA: A modelling and Simulation System for Sound and Image Synthesis – The General Formalism", *Computer Music Journal,* 17(4), 1993.

[13] B. P. Lathi, *Signal Processing and Linear Systems,* Berkeley-Cambridge, 1998

[14] C. Cadoz, "The Physical Model as Metaphor for Musical Creation "pico..TERA", a piece entirely generated by physical model", *in Proceedings of the 2002 International Computer Music Conference*, International Computer Music Association, Gotenborg, 2002

[15] C. Cadoz and J.L Florens, "The Physical Model : Modeling and simulating the instrumental universe", In G. De Poli, A Picall, and C. Roads (eds), *Representation Of Musical Signals*, pp. 227-268, Cambridge, MIT press, 1991

[16] N. Castagne, C. Cadoz, "Creating music by means of 'Physical Thinking': The Musician oriented GENESIS environment", in *Proc. Workshop on Digital Audio Effects (DAFx'02),* Hamburg, Germany, 2002

[17] Philip M. Morse, K. Uno Ingard, *Theoretical Acoustics*, Princeston University Press, 1986

[18] G. C Temes, J. W. LaPAtra, *Introduction to Circuit Synthesis and Design,* McGraw-Hill, 1977

[19] L. P. Huelsman, *Active and Passive Analog Filter Design,* McGraw-Hill, 1993

[20] G. H. Tomlinson, *Electrical Networks and Filters – Theory and Design,* Prentice Hall, 1991

[21] A. Kontogeorgakopoulos, C.Cadoz, "Digital Audio Effects and Physical Modelling", in *Proceedings of 2005 Sound and Music Computing Conference,* Salerno, Italy, 2005

[22] E. Incerti , "Synthèse de sons par modélisation physique de structures vibrantes : application pour la création musicale par ordinateur", *PhD Institut National Polytechnique de Grenoble*, France, 1993

[23] J. O. Smith, "*Physical* Modeling Using Digital Waveguides", *Computer Music Journal* 16(4), 1992

Author Index