MATCHING LIVE SOURCES WITH PHYSICAL MODELS

Paul Brossier, Mark Sandler, Mark Plumbley

Digital Music Lab.
Dpt. of Electronic Engineering
Queen Mary, Univ. of London
paul.brossier@elec.qmul.ac.uk

ABSTRACT

This paper investigates the use of a physical model template database as the parameter basis for a MPEG-4 Structured Audio (MP4-SA) codec. During analysis, the codec attempts to match the closest corresponding instrument in the database.

In this paper, we emphasize the mechanism enabling this match. We give an overview of the final front end, including the pitch detection stage, and remaining problems are discussed. A draft implementation, written in the Python language is described.

1. INTRODUCTION

Traditional coding systems for non-speech audio, such as MPEG-Audio [1] often use transform coding or filterbank methods, obtaining reduction in bit-rate by exploiting the masking properties of the human hearing system. For very low bit rate audio coding, important where very limited bandwidth is available or bandwidth is expensive, such transform coding is no longer sufficient. For example, it is currently not possible to transmit high quality audio in real time over a standard dial-up modem or at bandwidths normally available to a mobile device.

An alternative approach is to use the concept of *object-based audio coding*. The idea here is to encode the audio scene as a set of audio objects, together with a description of how to render those objects. This approach is embodied in the recent MPEG4 Structured Audio (MP4-SA) standard [2] which has the potential to provide a very compact representation of simple audio scenes [3].

MP4-SA allows any sound to be synthesised by sending a definition of a set of sound-producing objects, followed by instructions describing when and how those sound-producing objects are to be played. The *orchestra* (the sound-producing objects) is sent in SAOL (Structured Audio Orchestra Language), followed by a *score* (playing instructions) in SASL (Structured Audio Score Language). MIDI is built into the MP4-SA definition as a special type of score for convenience. Therefore if we have access to instrument definitions and score (or MIDI) for a musical audio signal, we can construct a very compact encoding of the signal which will be rendered by an audio synthesiser at the receiver.

However, if we do not already have the instrument definitions and score, we must extract this from the audio signal itself. This is a very difficult task for general musical audio signals, related to the problems of computational auditory scene analysis and automatic music transcription [4].

For our research, we are concentrating on real-time delivery of audio to devices with limited bandwidth and computational capability, such as mobile or embedded devices. We are therefore starting with simple audio scenes, considering how these can be analysed, transmitted and rendered in real time. We assume that the analysis stage is allowed to be more complex than the rendering device, since this would be performed at the content generator, where more computational capacity will be available.

In a previous paper [5], we reported an approach to object-based coding based on the extraction of harmonics. This was a development from the HILN (Harmonic and Individual Lines plus Noise) coding scheme [6], which has also been proposed for low bit-rate audio coding. This scheme allows *scalable coding*, whereby the number of parameters transmitted and processed can be adapted to match both the bandwidth required and computing load at the receiver [5].

For this paper we take an alternative approach. Instead of combining harmonic lines into objects, we will attempt to find the parameters of a physical instrument model that might have produced the given audio signal. Perhaps the simplest of this type of model is the *damped sinusoid*: a single-pole resonator excited by an impulse [7].

2. PHYSICAL MODELING

Physical models, and more generally source-filter models, have first been used for voice synthesis. The glottal pulse train, the source, is exciting the vocal tract, which acts as a filter. Early works on musical instruments include [8]. The last few years have seen significant advances in physical model synthesis [9], and in particular in computationally efficient models of synthesis based on waveguides. Physical models now offer faithful rendering of several real instruments and have recently been used in commercial synthesisers. Recent advances brought more interactivity to the models, allowing a performer to control "virtual acoustic" synthesiser with a high level of expression. Unlike sampling synthesis, object based synthesis models require far less memory, though more computational power.

Because of their linearity, string instruments are relatively simple to model. Generally, the wave equation is found as the solution of the differential equation governing the acoustical instrument. The direct computing of these equation is often prohibitively expensive for real-time performance. However, numerous models, such as string instruments, are accurately modelled with simplified equations, such as one-dimensional waveguides. Those waveguides are themselves modeled as bidirectional delay-lines with a very low computational cost.

For instance, the *commuted waveguide synthesis* algorithm [10, 11] provide an efficient scheme for various strings at a very low cost. Fig. 1 shows the block diagram of the *commuted waveguide*

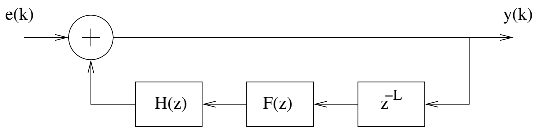

Figure 1: Commuted waveguide synthesis string model

synthesis algorithm. Its transfer function can be written as:

$$S(z) = \frac{1}{1 + z^{-L}F(z)H(z)}$$

where z^L is an integer delay, $F(z)$ a fractional delay and the *loop filter* $H(z)$, implemented as a one pole low pass filter:

$$H(z) = g\frac{1+a}{1+az^{-1}}$$

In this example, the delay L can be assigned to tune the fundamental frequency f_0 of the string: we have $L = f_s/f_0$, f_s being the sampling frequency. The fractional delay is used to fine tune the fundamental frequency. The low-pass filter coefficients a and g are assigned to match the desired spectral content.

Nowadays, efficient end-user implementations are available and work in real time. Numerous instruments, such as strings but also some bow and wind instruments, are already well known and can be easily implemented. Moreover, physical modeling of musical instrument has the advantage of giving the synthesis parameters a musical meaning. In the following sections, we will investigate how to build such models and extract their parameters from the analysis of simple monophonic sources.

3. IMPLEMENTATION

3.1. Framework

The MPEG4-SA standards implements two types of clock: sampling and control clocks. The control rate (`krate`) is typically 40 times smaller then the sampling rate (`arate`). During the client initialisation, a template instrument is sent. Its parameters are evaluated at `krate` while the result of the instrument execution is then written at `arate` to the client output. Thus, after choosing the appropriate model and building the orchestra descriptor, the model can be re-synthesised at the receiver at `arate` and controlled by the server at `krate`.

3.2. Preliminary analysis

This section describes techniques used in the preliminary analysis stage. They are based on [12, 13] and have been described in [5] Extracted parameters, from which we will decide how to build the model, include :

- onset detection, based on a combination of the derivative and the log derivative of the input,
- transient components extraction,
- harmonic components, extracted after the transient and based on the phase
- temporal envelope for each of the peaks, evaluated on a short time window,
- noise component, as the residual noise left after transient and steady states removal

3.3. Choosing the database

The model database contains generators and processors. Generators are simple signals, such as pulse trains, square, sawtooth, sine waves and noise. Processors are either frequency filter or delay lines. For instance, we use the `lopass` and `fracdelay` on a wave table to create a simple plucked string model. When they are not already implemented as such in the SAOL language, they can all be written as simple functions. Numerous post processing function, such as spatialisation, are already available to enhance the final resynthesis.

A *source* will more likely consist of one generator, eventually filtered. It will then be sent to an envelope generator, before feeding the model *filter*. The filter is composed of several processors.

Associated to these modules, a set of predefined templates must be defined, acting as a set of *heuristics*. Each of those templates represents a typical model of a real instrument, made of sources and filters and typical parameter values. For instance, the sawtooth-like signal of the bow is associated with the string loop filter. Both are given appropriate parameters to produce a correct violin note.

Each of those templates has a mapping associated to. The mapping translates the SASL MIDI-like data and the SAOL instrument parameters to the actual opcodes parameters. This way, the instrument can be driven by the MIDI score and a small set of chosen parameters. For instance, the

One of the advantages of such an object base is that it can be used to create new sounds. New association of units, such as a bow exciting a wind instrument or lips exciting a cymbal, can be drawn. Moreover, a mixture of two or more models can be used to enhance the modelled signal.

3.4. Exploring the model space

Where to start to find the appropriate models ? Out of real time, one can imagine a system trying new models and learning from experience which new associations give the desired type of signal.

Within hard real-time, the choice of the model is mainly based on the harmonic content of the signal and the repartition of the energy along time and within a note. The closest known model is chosen with default parameters. Other parameters will then be tried to enhance the reconstruction of the original sound.

The fundamental frequency f_0 is deduced from the harmonic components and their weighting. This is done by a fast peak picking algorithm, from which the note pitch is chosen as the maximum likelihood.

Temporal envelops of each peak and transient shapes are used to determine the best template.

Onset detection is implemented using a subband hybrid approach described in [12] and is to be used to control the MIDI notes and their velocities (amplitude).

Transient extraction [13] is used to model the source estimated $e_{est}(k)$. Rather than using a Gaussian shaped white noise as the source input, the energy burst is coloured to match the transient. Time envelope and frequency content of the transient are used to control the noise colour. This allows to fit the modeling of each note's attack. Noise colouring has been shown to greatly improve the resynthesis [14].

3.5. Tuning the right parameters

In [15] Raphael describes the use of a Bayesian network for a real time accompaniment system. The system is learning from experience, by marking different tries as "hot" or "cold", depending on their level of relevance.

A similar method is being implemented in the presented system to track the change of non mapped parameters. A few sets of new parameters are given to the model. The parameter sets are marked as "hot" if the obtained sound is enhanced, "cold" if the result is worst. Only relevant "hot", often as few as two or three, are kept. The best match is sent to the synthesis engine.

Providing the *filter* is linear, a way of marking set of parameters as "cold" or "hot" is to inverse filter the original note. The original version of a previously analysed note is inverse filtered through the chosen filter. This yields to a source signal $e_{inv}(k)$. The *commuted waveguide synthesis* algorithm has been used in a similar manner but for audio restoration [16]. Parameters are marked hot if the error signal $e_{inv}(k) - e_{fil}(k)$ tend to be minimised.

3.6. Issues

Known issues include the build of new instrument templates, and a way of testing their robustness. The choice of the model during the first played notes after initialisation is also a key problem to address.

4. PRELIMINARY RESULTS

A program is being written to produce the MPEG-4 Structured Audio Orchestra Language [17]. The system is trained on original recordings to recognise between three different plucked string instrument models.

First results showed that the recognition system is able to determine the closest model in most of the cases. Estimation and following of the parameters gave good synthesis result, and the *learning* method provides significative enhancement of the model timbre.

5. CONCLUSION

A draft system for real time extraction of physical models from instruments recordings has been presented. This system aims to gives a very compact description of the signal being analysed, using source filter models.

Both versatility and robustness of this system need enhancement. Future research will look at extending the databases and optimising the parameter mapping.

6. ACKNOWLEDGEMENTS

PB is supported by a Department of Electronic Engineering Scholarship. This work is also partially supported by grant GR/R54620 from the UK Engineering and Physical Sciences Research Council.

7. REFERENCES

[1] K. Brandenburg and M. Bosi, "Overview of MPEG Audio: Current and future standards for low bit rate audio coding," *J. Audio Eng Soc.*, vol. 45, pp. 4–21, 1997.

[2] Eric D. Scheirer, "The MPEG-4 Structured Audio standard," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP-98)*, 1998.

[3] B. L. Vercoe, W. G. Gardner, and E. D. Scheirer, "Structured Audio: Creation, transmission, and rendering of parametric sound representations," *Proceedings of the IEEE*, vol. 86, no. 5, pp. 922–940, May 1998.

[4] M. D. Plumbley, S. A. Abdallah, J. P. Bello, M. E. Davies, G. Monti, and M. B. Sandler, "Automatic music transcription and audio source separation," *Cybernetics and Systems*, vol. 33, no. 3, pp. 603–627, Sept. 2002.

[5] Paul Brossier, Mark Sandler, and Mark Plumbley, "Real Time Object Based Coding," *Proc. of the AES 114th Convention, Amsterdam, The Netherlands*, 2003.

[6] Bernd Edler and Heiko Purnhagen, "Parametric audio coding," in *5th International Conference on Signal Processing (ICSP 2000), Beijing*, Aug. 2000.

[7] M. Goodwin and M. Vetterli, "Atomic decompositions of audio signals," in *Proc. IEEE Workshop on Audio Signal Processing*, 1997.

[8] James Woodhouse, *Physical models of bow instruments : Applications to the violin*, Ph.D. thesis, 1974.

[9] Julius O. Smith, III, "Physical modeling synthesis update," *Computer Music Journal*, vol. 20, no. 2, pp. 44–56, 1996.

[10] Julius O. Smith, III, "Efficient synthesis of stringed musical instruments," in *Proceedings of the International Computer Music Conference, ICMC 93*, Tokyo, Japan, 1993, pp. 64–71.

[11] M. Karjalainen, V. Välimäki, and Z. Jánosy, "Towards high-quality sound synthesis of the guitar and string instruments," in *Proceedings of the International Computer Music Conference, ICMC 93*, Tokyo, Japan, 1993, pp. 56–63.

[12] Christopher Duxbury, Mark Sandler, and Mike Davis, "A Hybrid Approach to Musical Note Onset Detection," *Proc. of the DAFx Conference, Hamburg, Germany*, 2002.

[13] Christopher Duxbury, Mike Davis, and Mark Sandler, "Separation of Transient Information in Musical Audio using Multiresolution Analysis Techniques," *Proc. of the DAFx Conference, Limerick, Ireland*, 2002.

[14] P. A. A. Esquef, L. W. P. Biscainho, and V. Välimäki, "Restoration and enhancement of solo guitar recordings based on sound source modeling," vol. 50, p. 227.

[15] Christopher Raphael, "Can the Computer Learn to Play Music expressively?," *Proc. of the 8th Int. Workshop on Artificial Intelligence and Statistics, Morgan Kaufman*, pp. 113–120, 2001.

[16] P. A. A. Esquef, L. W. P. Biscainho, and V. Välimäki, "Audio restoration using sound source modeling," in *Proc. 2001 Finnish Signal Processing Symp. (FINSIG'01)*, p. 47.

[17] Eric D. Scheirer, "SAOL: The MPEG-4 Structured Audio Orchestra Language," *Proc. IEEE*, 1998.