



# Audio Engineering Society Convention Paper 5809

Presented at the 114th Convention  
2003 March 22–25 Amsterdam, The Netherlands

*This convention paper has been reproduced from the author's advance manuscript, without editing, corrections, or consideration by the Review Board. The AES takes no responsibility for the contents. Additional papers may be obtained by sending request and remittance to Audio Engineering Society, 60 East 42nd Street, New York, New York 10165-2520, USA; also see [www.aes.org](http://www.aes.org). All rights reserved. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Journal of the Audio Engineering Society.*

---

## Real Time Object Based Coding

Paul M. Brossier<sup>1</sup>, Mark B. Sandler<sup>1</sup> and Mark D. Plumbley<sup>1</sup>

<sup>1</sup>*Queen Mary, University of London, Mile End Road, London E1 4NS, United Kingdom*

Correspondence should be addressed to Paul Brossier ([paul.brossier@elec.qmul.ac.uk](mailto:paul.brossier@elec.qmul.ac.uk))

### ABSTRACT

This paper describes the design of a real-time MP4 Structured Audio codec for monophonic signals. The coding of the live input consists of a pitch detection system which returns the MIDI-like data, and an additive synthesis scheme which creates and modifies the current instrument. Both parts are designed to be fast and scalable. The analysis parameters let the user choose the computational cost for both the analysis and the resynthesis. The extracted objects can then be used in live environments for encoding and/or creation.

### INTRODUCTION

The MPEG-4 standard is becoming increasingly common for low bit rate coding. Based on both Csound and MUSIC-N like languages, MPEG-4 Structured Audio (SA) provides a flexible framework for audio analysis and synthesis.

However, most of the MPEG-4 environments lack

real time capabilities, whereas many applications require real time object extraction: broadcasting at very low bit rates on any standard device, fast harmonic structure analysis and audio effects just to mention some.

In this paper, we present the first draft of a system designed to achieve this for monophonic signals (e.g.

single voice).

## AUDIO OBJECTS

### MPEG-4 Structured audio

The MPEG-4 Structured Audio standard [1] provides an object oriented framework for developing such a real time application. The instrument are described in the Csound-like Structured Audio Orchestra Language (SAOL) while the score is passed as MIDI or Structured Audio Score Language data (SASL).

The MPEG-4 standard has been designed to be widely portable. It is being integrated in various handheld or portable devices. DSP chipsets embedding a fully compliant MPEG-4 decoder will soon be available.

Including formats similar to MIDI and SoundFont, it can be seen as the next generation of Csound. It has recently been implemented in recent multimedia players (e.g. QuickTime [2]). Version 2 of the MPEG-4 SA standard [1] now includes the parametric audio coding tools Harmonic and Individual Lines plus Noise (HILN). Real time decoding of the HILN MPEG-4 bitstream is already available [3].

However, most of the MP4-SA interpreters currently lack real time encoding.

### Definition

We can differentiate three major scales of audio objects. The largest one would contain objects more global than a note, such as a musical sentence (melody), the tempo, the time signature and other entities relevant for the whole music piece. The middle range scale includes notes, amplitude envelopes. The smallest range would correspond to objects smaller than a note: transients, partials, tremolos, and so on.

Extracting such objects in real time brings up the not so obvious problem where we have to decide the identity of an object before reaching its end.

### Real time

In live situations, real time object extraction has to be done without knowledge of the future audio stream. Our approach is to send the objects we know as soon as they are identified. There are three rates at which the SAOL scripts are executed: they determine how fast the objects can be sent and up-

dated. The a-rate is the samplingrate (e.g. 44100 Hz). The k-rate is the control rate (e.g. 200 Hz for instance), and i-variables are updated every time an instrument is instantiated.

Neglecting network delays, the latency between the audio stream and the resynthesis should remain constant, and if possible as small 50 ms (or less) for performing.

Some other problems specific to streaming have been investigated in [4]. Amongst them are the lack of flexibility of the process and the difficulty to guarantee the reproduction quality of the decoder.

## OVERVIEW

### Architecture

The system architecture is divided in to main parts: the *server*, which does the analysis, and the *client*, which does the synthesis. The audio objects are passed from the server to the client using the MP4 bitstream format. Fig. 1 shows a diagram of the server part.

The analysis consists of extracting objects from the live input, typically the signal coming from the microphone of an orchestra instrument. The objects can be of any type, such as tempo, notes, transients. Since the end of the objects cannot be precisely defined, we need to guess what object we are dealing with to send it as soon as possible to the client. Therefore, the remaining parameters describing the object need to be stored in a buffer on the server side.

### Learning buffer

The *learning buffer* on the server side is used to store the extracted information. The client will receive the objects as soon as possible after they occurred, and will use the previous description of that object to rebuild it. At that time, the copy of the *learning buffer* on the client side is waiting to be updated.

The process is similar to the one described in [5]. In case the note has not been encountered yet, the analysis is stored on the server buffer, waiting to be sent towards the client. If the appropriate note description has not been sent yet, the synthesis is done by linearly interpolating the parameters from the already received ones. As soon as the remaining bandwidth is large enough, this new note is then sent

to the client and added to the previous instrument definition.

### Automation and adaptation

In order to handle various situation, we want to be able to adjust the computational load for both analysis and re-synthesis.

The analysis level on the server side can be set by adjusting the frame size and the available bandwidth.

While the synthesis client load should be scaled very low for handheld devices, it could be scaled much higher as well to enable high resolution re-synthesis when the client is on more powerful machines.

The frequency at which the instrument is updated depends on the available bandwidth. The various instrument peaks are regularly sent along the MP4 bit-stream. A very low bandwidth client could use a simple pure tone as a basic instrument, whereas a broadband system could receive the residual noise missed by the instrument model or the whole notes samples as raw data. This ensures the flexibility of the system. Analysis and synthesis parameters might be set to run on any MP4-SA compliant device. The quality of the obtained bitstream will depend on the available resources at each side.

### Streaming

Updating the SAOL instrument remains a major issue, as it has already been received and instantiated. In spite of using a new instrument, the chosen option is to use wavetables on both sides for storing instrument parameters: frequencies, phases and amplitude envelopes. The instrument parameters are thus updated as soon as the wavetables have been received.

## IMPLEMENTATION

In this first implementation, we use a phase vocoder [6] to do the analysis on the server side. The client performs the synthesis by using an oscillator bank.

The analysis consists of extracting the phases, amplitude and frequencies of a finite number of partials. From this data, a note decision is taken, and the nearest corresponding MIDI note is passed along the MP4-SA bit-stream as a SASL script. The SAOL instrument uses a fast additive sinusoidal model to resynthesise the note. Improvements and updates are brought to the instrument while the following notes arrive, and the modified instrument is passed

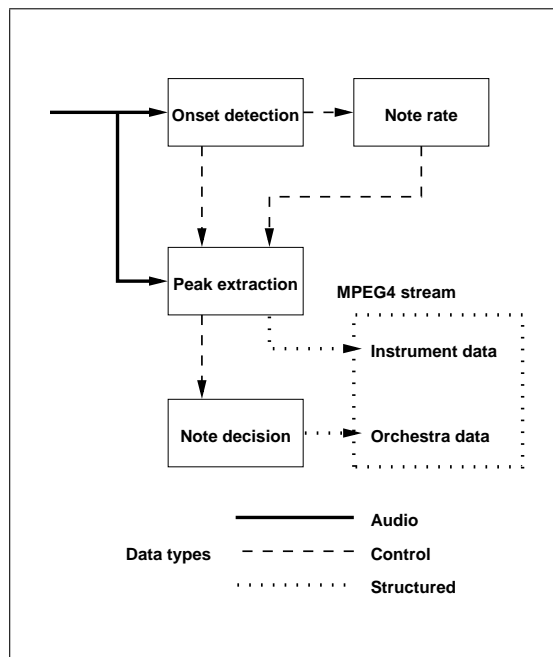


Fig. 1: Server Overview.

along the MP4-SA bit-stream. If the client receives a MIDI note for which the description has not been received yet, a linear interpolation is done from the nearest known waveform.

This implementation is somewhat similar to the HILN [7], except it uses SASL standard to pass MIDI-like data to the client, and has been optimized for real time operation.

### Peaks extraction

Because we want to extract more information than the single pitch, the autocorrelation method would not be appropriate for this system. However, we do need a fast algorithm. The matching pursuit method used in the MPEG-4 HILN codec uses a greedy algorithm which uses recursive subtraction to remove each identified component [7]. This requires more than one FFT per partial at every analysis instant.

Using the SAOL language, we have been writing a simple and fast pitch tracker, similar to the M. Puckette algorithm [8] based on the harmonicity of the spectrum. For each analysis instant, we calculate the Short Time Fourier Transform (STFT) of the signal around this instant. The peaks are selected straightforwardly after the STFT is computed, using

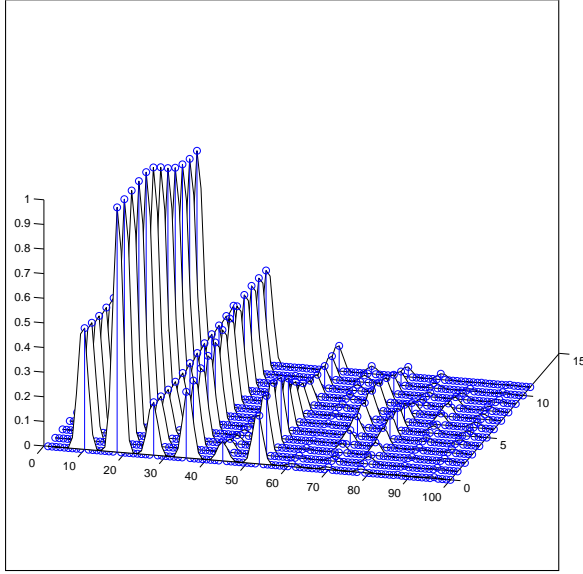


Fig. 2: Peaks extraction

a basic selection algorithm.

Fig. 2 shows an example of peak picking over several consecutive frames. The peaks are sorted according to their amplitude. The fundamental is chosen based on both harmonicity and amplitude, and a harmonic factor is used to describe the other partials. This factor can be any real number between 0 and the number of bins in the STFT. This way, inharmonicity will be described properly.

### Onset detection

To enhance note decision, the analysis algorithm also implements an optional energy based onset detection. It is a simplified version of the hybrid algorithm described in [9]. We use a constant-Q conjugate quadrature filter bank to split the signal into five subbands.

For the two lowest bands, onsets are determined by first calculating a distance measure between the present and previous frames :

$$DM = \frac{\sum_k (X_k(nh) - X_k((n-1)h))^2}{\sum_{k=1}^{N/2} |X_k((n-1)h)|^2} \quad (1)$$

where  $X_k(n)$  is the Fourier analysis in the bin  $k$  at time  $n$ .

For each higher subband signal, the onset detection

function is :

$$ons(n) = SE(n) - \sum_{a=1}^A \frac{SE(n-a)}{a} \quad (2)$$

where  $SE(n)$  is the subband energy given by:

$$SE(n) = \sum_{m=(n-1)h}^{nh} |x(m)|^2 \quad (3)$$

and  $A$  is the amount of frames over which the detection function is evaluated.

Onsets are determined as a combination of the detection function of each band.

This feature has been kept optional because of its high computing expense on the server side. However, the load is unchanged on the client side, while it allows us to dramatically enhance the time resolution.

### Scalability

A straightforward way to adapt the load on both sides is to change the number of peaks. This would mainly change the load on the client side, acting directly on the re-synthesis quality.

Changing the delay between analysis times is also a good way of scaling the client load. Moreover, at a fast note rate, small frames are required to enhance onset detections and separate rapid changes. On the other hand, wider windows can be used to enhance frequency resolution.

In order to keep the computation load constant, the number of bins in the STFT is set to be proportional to the analysis window length. Therefore, we want to find the right trade-off between time resolution and frequency resolution.

In any case, the number of peaks and the time resolution need to be constant. But by using the onset detection method described above, we can adapt the analysis frequency according to the resulting note rate. The frequency resolution is thus improved for low tempi and the time resolution will be enhanced at fast note rates.

### Transient separation

For better resynthesis, separation between transient and steady state has been investigated. The algo-

rtihm is based on [10], which looks for fast phase changes in each bin. A threshold  $T_t$  for separating transient from steady states is set. Transients are detected if the following condition holds:

$$\phi_k(n) - 2\phi_k(n-1) + \phi_k(n-2) > T_t \quad (4)$$

Where  $\phi_k(n)$  is the phase in the bin  $k$  at the analysis time  $n$ . The selected bins contain the transient part of the signal. They can thus be sent to the client as complementary information at every detected onsets. The resulting re-synthesis is considerably improved as the note attacks (e.g. onsets) have better frequency and time resolution.

## RESULTS

As HILN, this coding is not meant to provide a high quality coding, but to extract audio objects and send them through very low bandwidth channels. Subjective audio quality of the HILN encoding has been evaluated in [3]. However, the transient extraction feature is very promising.

The Current SAOL algorithm occupies approximately 20% of the processor on an 700 MHz i686 processor, for both analysis and resynthesis of 64 peaks. Whereas the load remains almost constant with fewer peaks, a large number of peaks cause some erroneous data to be sent and thus results in unwanted noise. A good optimization should allow one to reduce this cost by a factor from 5 to 10.

## CONCLUSION & FURTHER WORK

In this paper, we have described an approach to real time object based coding based on MPEG-4 SA. A first version, written in SAOL, yields promising results and suggests a robust real time object system.

A comparison with HILN codec has to be done in order to verify the robustness of the analysis part and to compare computational cost.

Amongst the further directions for this research are study of polyphonic signal and evaluating the robustness of the above proposed implementation to separate different notes of a chord.

An interface will be written to control the analysis on the server side. Recent libraries such as CLAM will be used, giving the user interactive ways of setting

the analysis parameter. Another GUI is also needed for client post processing.

## ACKNOWLEDGEMENTS

PMB 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.

Thanks to Juan Pablo Bello and Chris Duxbury for their comments and suggestions.

## REFERENCES

- [1] ISO/IEC, 14496-3 FDAM1: MPEG-4 Audio Version 2, ISO/IEC JTC1/SC29/WG11 N3058, Dec. 1999.
- [2] Quicktime developer website, *What's New in QuickTime 6*, <http://developer.apple.com/quicktime/>, 2002.
- [3] H. Purnhagen, N. Meine *HILN - The MPEG-4 Parametric audio coding tools*, Proc of the ISCAS, Geneva, Germany, 2000.
- [4] M. Claussen and L. Solbach, *Streaming Structured Audio*, Proc. of the ICMC, Berlin, Germany, 2000.
- [5] J. P. Bello, L. Daudet, and M. Sandler, *Time-domain Polyphonic Transcription using Self-Generating Databases*, Proc. of the AES 112th Convention, Munich, Germany, 2002.
- [6] M. Dolson. "The phase vocoder: A tutorial". *Computer Music Journal*, 10(4):14-27, 1986.
- [7] H. Purnhagen, N. Meine *Speeding up HILN - MPEG-4 Parametric Audio Encoding with Reduced Complexity*, Proc. of the AES 109th Convention, Los Angeles, 1999.
- [8] M. Puckette, *Real-time audio analysis tools for Pd and MSP* Proc. of the ICMC, Ann Arbor, Michigan, USA, 1998.
- [9] C. Duxbury, M. Sandler and M. Davis, *A Hybrid Approach to Musical Note Onset Detection*, Proc. of the DAFx Conference, Hamburg, Germany, 2002.

- [10] C. Duxbury, M. Davies, M. Sandler, *Separation Of Transient Information In Musical Audio Using Multiresolution Analysis Techniques*, Proc. of the DAFx Conference, Verona, Italy, 2001.