Algorithmic and Wavetable Synthesis in the MPEG-4 Multimedia Standard

ERIC D. SCHEIRER, *AES Student Member*

MIT Media Laboratory, Machine Listening Group, Cambridge MA

AND

LEE RAY

Joint E-Mu/Creative Technology Center, Scotts Valley CA

The newly released MPEG-4 standard for multimedia transmission contains several novel tools for the low-bitrate coding of audio. Among these is a new codec called "Structured Audio" that allows sound to be transmitted in a synthetic description format and synthesized at the client. MPEG-4 Structured Audio contains facilities for both algorithmic synthesis, using a new software-synthesis language called SAOL, and wavetable synthesis, using a new format for the efficient transmission of banks of samples. We contrast the use of these techniques for various multimedia applications, discussing scenarios in which one is favored over the other, or in which they are profitably used together.

1 INTRODUCTION

The MPEG-4 audio standard [6], formally known as ISO/IEC 14496-3, will be completed in October 1998 and published as an International Standard in December 1998. Audio coding in MPEG-4 extends the sound-coding model in previous MPEG standards in two ways. First, techniques are specified for low-bitrate lossy compression of sound [14]; second, MPEG-4 integrates the transmission of natural (compressed) sound and synthetic sound.

In the synthetic-sound coding part of the standard, called Structured Audio [16], sound is transmitted through one of several *sound specification* formats, in which sound is not compressed but described in the bitstream. The sound description is processed at the receiver and real-time sound synthesis is used to produce sound. Structured Audio techniques allow sound to be coded and transmitted at very low bitrates, since this structural description of a sound can be many times more compact than the sound created upon synthesis.

There are two main modes of operation of MPEG-4 Structured Audio. The first is an efficient coding format for wavetable (sampling) synthesis, in which sound is described as the sum over time of a number of pitch-shifted and filtered audio samples. This method is commonly used today in sound cards and in software, and the format unifies the capabilities of several existing *de facto* standards. The second mode is a format for general-purpose software synthesis, in which sound is described using a powerful algorithmic

description language. In this method, sound-synthesis and sound-processing algorithms are transmitted in the bitstream and used to create sound at the receiver. Techniques similar to this have been used in the computer-music-research field for many years, but MPEG-4 Structured Audio represents their first standardization as audio-coding technology.

In this paper, we focus from an applications standpoint on the uses of these two methods for the very-low-bitrate transmission of sound. We will contrast the use of wavetable and algorithmic synthesis in various applications, describing the advantages and disadvantages of each technique. We will also present a discussion of hybridized audio synthesis in MPEG-4, in which elements of both the wavetable and algorithmic toolset are used. We will relate the very-low-bitrate coding techniques possible with MPEG-4 Structured Audio to the low-bitrate coding of natural sound in MPEG-4, and discuss new prospects for hybrid low-bitrate coding which makes use of concepts both from traditional compression and Structured Audio coding. The paper concludes with a discussion on the current state of implementation of the standard.

2 MPEG-4 STRUCTURED AUDIO

Other references [16], [18] describe the structure and capabilities of the MPEG-4 standard for Structured Audio in detail; this section provides an overview to lend context to the material that follows.

MPEG-4 was conceived as a standard to enable low-bitrate coding over digital radio, the Internet, and other links which do not support the multi-megabit rates needed to allow high-quality video transmission. Over its progress toward standardization, these capabilities expanded somewhat; the result is a unified standard for interactive, distributed, low-bitrate, scalable and object-based multimedia. A key concept in MPEG-4 is Synthetic-Natural Hybrid Coding, or SNHC. SNHC techniques in MPEG-4 allow the juxtaposition and composition of natural (compressed) audio and video with synthetic sound and visuals created in real-time on the rendering terminal.

MPEG-4 Structured Audio is the mechanism which allows SNHC audio coding techniques to be used. The Structured Audio toolset provides advanced capabilities for the transmission of synthetic soundtracks, and for the mixing and software-based postproduction of soundtracks which use both natural and synthetic audio. With these tools, hybrid synthetic/natural soundtracks are transmitted as a number of *sound streams*, perhaps using a different coding tool for each. The sound streams are mixed and post-processed to create *sound objects*. The sound objects may then be given 3-D spatial locations, attached to visual objects in the scene, and dynamically and interactively moved about the presentation space. The advanced-effects capabilities of MPEG-4 Structured Audio are described in [18].

Synthetic sound is described in MPEG-4 with two main tools. The music language SAOL [17], which stands for “Structured Audio Orchestra Language” and is pronounced “sail”, is used to specify sound-processing and sound-synthesis algorithms to be executed in the terminal. It is a robust bitstream specification for *algorithmic* synthetic sound. In addition to, or instead of, SAOL, a rich wavetable format called SASBF, for “Structured Audio Sample Bank Format”, may be used to transmit banks of sound-sample data to be used in wavetable, or sampling synthesis. Both of these methods may be controlled with MIDI instructions [12] or with a new control format called SASL, for “Structured Audio Score Language.”

The remainder of this section discusses the sound-synthesis capabilities of MPEG-4 in more detail.

2.1 Algorithmic synthesis with SAOL

SAOL is a powerful and flexible music language of the type known as “Music-N” languages. Other languages of this type are Csound [20], Nyquist [3] and SuperCollider [10]. SAOL is not a method of sound synthesis, but a language for the description of methods of sound synthesis. Any existing or future sound-synthesis algorithm may be described in SAOL.

The bitstream format for Structured Audio bitstreams using SAOL is similar to that of other audio coders: it begins with a *configuration header* that describes details about the streaming data to follow, and then frames of streaming data. The configuration header is somewhat more complex than other audio coding methods, though: it contains a description of several sound-synthesis algorithms which will be used to create the sound. When the session starts, the header is parsed by the receiving terminal and used to configure a special synthesis engine. This synthesis engine is capable of being reconfigured to handle many different kinds of sound algorithms, including wavetable, FM, physical-modeling, granular, additive, and subtractive synthesis, and custom hybrids of all of these.

SAOL, like the other synthesis languages mentioned above, is built around the *unit generator* paradigm for synthesis. Synthesis algorithms are described as the interaction of primitive sound-generators which may be banked in series or used in parallel to create sounds.

The bitstream data in a Structured Audio stream contains control information that drives the synthesizers transmitted in the header. The control information may be expressed in MIDI, which allows note-on, note-off, and simple control messages to be transmitted. A new format called SASL is also standardized. SASL allows more precise and flexible control of synthesis than MIDI, including double-precision time-stamps and control values, and arbitrary mapping between controllers and their effect on the sound.

2.2 Wavetable synthesis with SASBF

The SASBF wavetable-bank format had a somewhat complex history of development. The original specification was contributed by E-Mu Systems and was based on their "SoundFont" format [15]. After integration of this component in the MPEG-4 reference software was complete, the MIDI Manufacturers Association (MMA) approached MPEG requesting that MPEG-4 SASBF be compatible with their "Downloaded Sounds" format [13]. E-Mu agreed that this compatibility was desirable, and so a new format was negotiated and designed collaboratively by all parties.

The resulting format is based on the DLS standard; however, several improvements allow it to have better sound and functionality than a bare-bones DLS implementation. Sound is coded by transmission of *banks* of wavetable data (samples). These banks hold PCM data which is critically sampled and organized by the content author to represent the instruments needed for a particular composition. As well as the banks of samples themselves, various parameters may be transmitted which control the way the various pitches (*notes*) are mapped into samples, as well as envelopes and dynamic resonance filters which apply to the samples. Synthesis based on SASBF is performed by pitch-shifting, filtering and mixing the samples to result in a desired sound.

As compared to the DLS standard itself, the format is less extensible and more directed about which elements are required. That is, where there is freedom to make certain additions and improvements to a basic DLS system, this freedom is not part of the normative MPEG-4 specification. The format may be viewed, in fact, as one particular instance of a family of formats describable with DLS, which is a very general format. The careful standardization of elements, though, means that content developers have a more powerful feature set and a better guarantee of compatible cross-system performance.

It is a positive result for the standardization effort that negotiation regarding wavetable standards has resulted in a mutually-agreeable format. The existance of multiple popular but mutually incompatible formats has been a source of consternation in the content community for some time; with the harmonization and standardization of SASBF, content and hardware development efforts may proceed in harmony rather than operating at cross-purposes.

The most usual way to control SASBF is using MIDI control data; the format, like DLS, is designed for this purpose. However, in a full-featured MPEG-4 Structured Audio system, both SAOL functionality and SASBF functionality are available. In this scenario, the SASBF synthesis is executed under the control of the downloaded SAOL instrument code, and the resulting sound is available for further processing in SAOL. When this mechanism is used, either SASL or MIDI may be used to control synthesis.

Thus, for example, a content author may use the SASBF format to efficiently transmit banks of samples, but use SAOL to transmit unusual reverbs, resonators, or other post-filters which apply to the result. Further, the exact dispatch of SASBF synthesis is not limited to the simple way that is allowed with MIDI, but more sophisticated layering and control is possible.

2.3 Client-side postproduction with AudioBIFS

The AudioBIFS format in MPEG-4 [18] enables the client-side mixing and synchronization of sound objects. AudioBIFS is part of the MPEG-4 BIFS (Binary Format for Scene Description) tool, which allows scene-graph based organization of synthetic and streaming content. Using AudioBIFS, different parts of a sound scene may be coded in different ways. For example, in a soundtrack which requires speech voice-over with background music, the speech may be coded using a low-bitrate speech coder [14] and the music with the Structured Audio coder. These two elements are decoded separately and then mixed in software after they are received, before final presentation to the listener.

As well as simple mixing, more sophisticated effects processing is possible with AudioBIFS. Using the language (SAOL) used to describe sound-synthesis, arbitrary post-production algorithms may be downloaded and applied to natural or synthetic sounds. For example, to transmit the sound of a person speaking in a reverberant environment, the flat unreverberated speech is transmitted using the MPEG-4 speech coder, along with an algorithm implementing a synthetic reverbator [4]. The resulting sound is a higher-quality reverberated speech sound than could be acheived with the speech coder alone at that bitrate.

Using AudioBIFS also allows sound objects, once mixed and post-processed as described above, to be placed in 3-D space and synchronized with computer graphics or streaming video. The *scene graph* describing the position and synchronization of media objects may allow user interactivity, so that different sounds, images, motion, or post-production may occur in response to the user's actions. MPEG-4 BIFS is a very general format allowing powerful client-side manipulation of multimedia scenes.

2.4 MPEG-4 Structured Audio Profiles

Profiles are defined in the MPEG-4 standard to allow the construction of interoperable subsets of the standard. By making a system compliant to a profile of MPEG-4, a manufacturer may be guaranteed that a limited range of content will be playable, without the expense of requiring a whole MPEG-4 implementation in the system.

There are three Profiles for Structured Audio transmission in MPEG-4. In the most flexible, fully-capable SAOL, SASL, SASBF, and MIDI implementations are required. Using this Profile (the "Full Profile"), very sophisticated algorithmic and hybrid synthesis may be accomplished. Another Profile, the "Wavetable Synthesis Profile", allows only the transmission of SASBF data and MIDI files. This Profile provides functionality approximately equivalent to today's commonly-available sound peripherals, but allows these soundtracks to be incorporated in MPEG-4 content (for example, synchronized with singing voice tracks). Such a Profile has wide applicability in the karaoke and other interactive-entertainment markets. However, the advanced-effects capabilities of AudioBIFS are not available in this Profile.

Finally, a simple Profile, the "General MIDI Profile", allows only MIDI data to occur in the bitstream. In this profile, the semantics (decoding instructions) of the bitstream are those standardized for the General MIDI standard by the MIDI Manufacturers Association. Unlike the two above Profiles, in the General MIDI Profile, the sound quality is not normative, and so content authors are not guaranteed to have high sound quality in this profile. However, very inexpensive decoders using ASIC-based GM devices, a commodity resource today, are possible.

3 TRADEOFFS BETWEEN ALGORITHMIC AND WAVETABLE SYNTHESIS

The remainder of the present paper discusses the various uses of algorithmic and wavetable synthesis in applications which make use of MPEG-4 functionality. Unlike other methods for transmission of synthetic sound, the MPEG-4 Structured Audio tools are rich and varied, and do not lock the musician into a

particular way of creating content. Many scenarios can make use of only a particular Profile of the full toolset, or only a subset of the capabilities available in the Full Profile. As fully-capable MPEG-4 implementations become available, content authors and sound designers will have to engage in careful analysis to determine the particular way to best make use of the different functions for their application.

3.1 Applications for General-MIDI synthesis

We will not discuss the General MIDI profile to any significant extent. This profile, while important for allowing the use of very simple decoding-rendering systems, is not appropriate for serious content due to the lack of control over sound quality. The scenarios in which General-MIDI transmission of musical content is appropriate are limited to the simplest of audio icons, jingles, and musical backgrounds, or those in which the cost of the rendering device must be extremely low.

3.2 Applications for wavetable synthesis

The use of rich wavetable formats to describe music and sound effects has increased dramatically over the past year. Formats such as MIDI-DLS [13], Rich Music Format [7], and Creative SoundFonts [15] are used today to transmit music over the Internet, and to communicate between applications such as videogames and sound hardware with wavetable-synthesis capabilities. Recent developments in operating-systems design [11] embed the ability to transfer data in this manner directly into the audio subsystem of the operating system.

With the development of MPEG-4, these capabilities will be used not only in PC-based communication such as gaming and Internet communications, but in traditional broadcast settings, including digital radio and satellite television. The strengths and weaknesses of this method, as compared to algorithmic synthesis, are somewhat different in such scenarios.

3.2.1 Bandwidth considerations

Most obviously, in transmission of high-quality sound through wavetables, the bandwidth required is greatly *front-loaded*. That is, a great deal of data (the wavetable sample bank) is transmitted at the beginning of the session, in the stream header, and after that only a small amount of streaming data needs to be transmitted to describe the use of the wavetables. At a minimum, only note-on, note-off information is absolutely necessary; such data only represents a data rate of 10-250 bps for most music. The use of continuous controllers augments this data rate somewhat, but with judicious use even the most expressive music describable with MIDI will only utilize an average bitrate of less than 1 kbps.

The primary disadvantage of the front-loaded aspect of SASBF description is that, for broadcast sessions, it makes “tune-in” random access difficult or impossible with high-quality music. This sort of random access is required for one-way broadcast applications, in which listeners wish to randomly tune a digital radio to a station. It is not possible to decode the SASBF stream properly without access to the sound samples in the header, because it is only there that the actual “sound” of the content is defined. In order to make this tune-in possible, the stream header information must be continuously rebroadcast – tune-in is only meaningful when the SASBF header has already been transmitted.

For many high-quality programs, the sample bank is relatively large – it is not unusual for a high-quality bank of samples for a pop tune to be 1 MB or larger in size. Continuous rebroadcast of such information is prohibitively expensive. Thus, high-quality wavetable synthesis cannot be used in one-way broadcast applications.

This consideration can be relaxed somewhat in one of two ways. First, *back-channel* functionality makes tune-in access much easier. If the communications path is not strictly one-way, then the receiver may signal to the server/sender when it wishes to be configured for a new piece of program material. Only at this time, for this receiver, does the configuration information need to be rebroadcast. A specification for back-channel communication is provided in the MPEG-4 standard.

Second, receiver-side *caching* of standard sample banks can enable a particular content provider and receiver manufacturer to signal that a needed sample bank is already present at the terminal. This data may be downloaded the first time a particular "show" is received, and then saved, or it might ship preinstalled on the receiver hardware. MPEG-4 does not currently provide a formal specification for receiver-side caching, but it may be easily implemented in a terminal-specific way and signalled using the existing private information components of the bitstream. A back-channel can also aid this implementation, by allowing the server to query the receiver as to what cached data is available.

If neither back-channel nor caching functionality are available, then the only way that SASBF data is used in a program session over low-bitrate broadcast is with very small samples. It is possible to use loops on very short samples to create sound, but difficult to create high-quality sound in this way. This is because the perceived "richness" of acoustic performances arises from the complex long-term modulation of the timbres of acoustic instruments. If long samples are used, this modulation is effectively captured and used in synthesis, but with short samples, the long-term spectral and pitch shifts must be recreated algorithmically, which is difficult.

3.2.2 Complexity considerations

The primary advantage of wavetable synthesis is that it provides bounded and easy-to-analyze computational complexity. The resources required to synthesize sound from wavetables are well-understood today, and inexpensive hardware solutions on ASICs are widely available from numerous vendors. Thus, if the restricted wavetable profile of MPEG-4 is used for an application, the receiving terminals can be built relatively easily and inexpensively. Except for the capabilities which allow wavetable-synthesis to be mixed with decoded natural audio streams, the functionality required is essentially the same as today's PC sound hardware.

Further, as compared to the full profile including algorithmic synthesis (about which see below), it is easier for content developers to understand the resources available to them. The complete specification of MPEG-4 *levels*, which provide bounds on terminal resources, is not yet complete, but levels for the Wavetable Synthesis Profile will likely be set simply by bounding the maximum number of simultaneous voices. Analysis of a particular piece of content in order to understand such a maximum is relatively straightforward.

3.2.3 Applications

From the above analysis, we can arrive at a set of primary applications for SASBF-based wavetable synthesis alone. Key application considerations for the use of SASBF include the following:

- Strong price pressure for decoding terminals
- Limited development resources for terminals and content
- If session tune-in capability is required, then low sound quality must suffice, or else local storage, existence of a back-channel, and/or high-bandwidth channel capability must be available.
- No requirement for high-quality client-side post-production and downloaded effects.
- No requirement for extreme musical expressivity.

Such applications today include karaoke systems, simple digital radios, musical backgrounds for WWW pages, and musical entertainment for children. The key capability provided by MPEG-4 in these contexts is the ability to synchronize and layer musical sound and human speech such as voice-overs or prompts, and the ability to synchronize sound with text and images.

3.3 Applications for algorithmic synthesis

Models of music description using general-purpose software have been available for many years. Composers, especially in academia, have used languages such as Music-IV [9], Csound [20], and SuperCollider [10] to compose music drawing from a wide range of sonic possibilities. The recent

development of using general-purpose software synthesis as a transmission protocol – first in the prototype "NetSound" system [2] and now in MPEG-4 – allows these techniques to be brought to bear in a more general audio-coding environment.

However, it is only recently that general-purpose software synthesis implementations have allowed real-time "live" synthesis. Previously, the dominant paradigm involved an off-line synthesis step, which might take many hours, before the synthesized musical sound was available for listening. The Structured Audio paradigm for sound transmission is a practical one today exactly because such a step is no longer necessary, and interesting synthesis can be effected in real-time on reasonably-priced hardware.

As it is a very general and flexible protocol for synthesis control, there are few fixed considerations regarding algorithmic synthesis in MPEG-4. Some bitstreams are highly complex to decode and some are not, some enable tune-in easily and some do not. Unlike with the SASBF format, the bitstream author takes more responsibility to ensure that a bitstream is appropriate for the terminals that it must play on; it is much harder to develop tools that automate such decisions for the Full Profile than it is for the Wavetable Profile.

3.3.1 Bandwidth considerations

Depending on the synthesis methods used, algorithmic synthesis may have similar properties to wavetable synthesis. Sound samples may be transmitted and used by SAOL for synthesis; wavetable synthesis is one type of synthesis that can be conveyed with SAOL algorithms. For any method of synthesis that uses sound samples, the size of samples will typically be greater than the size of the algorithms that describe their use, and then the considerations are similar to those described above.

On the other hand, SAOL algorithms may be extremely concise and yet highly expressive. High-quality physical modelling synthesis [19], for example, requires very short algorithms, and no samples, to implement. In such a case, periodic rebroadcast of the configuration header is not impossible, and so the tune-in problem is avoided. The simplest music (monophonic and not expressively controlled) can be transmitted at less than 1 kbps, even allowing continuous tune-in. If tune-in is not required, or a back-channel is available, many Structured Audio bitstreams can be transmitted at 10 bps or less, if sound parameters can be created algorithmically.

Other synthesis methods have a bandwidth characteristic that is more similar to that of traditional audio coding. Particularly with *analysis/synthesis* methods, in which a desired sound is analyzed, data-reduced, and streamed as a set of parameters to a particular sound model, sounds may be transmitted with a relatively concise header and a larger amount of streaming data. Techniques such as data reduction through linear prediction [8] (which is also the basis of CELP-based speech coding methods [5]) transform a sound into frames of data in an analysis stage, and then perform modifications on the data before resynthesis with a subtractive filtering process. Such techniques require somewhat more bandwidth – synthesis based on linear-predictive coding begins to have application at 1-2 kbps – but easily enable break-in, since most of the important information is in the streaming data.

Thus, where very-low-bitrate requirements primarily affect the sound quality of SASBF synthesis – since in very-low-bitrate environments, only very short samples may be used – in SAOL-based algorithmic synthesis, this requirement predominantly affects the flexibility of synthesis algorithms. In a very-low-bitrate transmission system, frame-based analysis/synthesis is impractical, large wavetables cannot be used, and extremely complex continuous control of sound generation is not possible. Instead, the achievable expressivity is limited to succinct algorithms with simple control. However, this is not so much a restriction on the sound *quality* as on the *types* of sounds [22].

Tune-in functionality can be more of a problem with Full Profile bitstreams, because the header (SAOL algorithms) alone may not be enough to allow correct decoding. For example, consider an SAOL instrument which sounds like a piano for the first note played on it, a drum for the second note, a piano for the third note, a drum for the fourth, and so on. Such an unusual instrument cannot be described in SASBF, but is easy to convey in SAOL code. If we wish to tune into the middle of a bitstream, though, it isn't

sufficient to know the synthesis algorithm and which note to play next – we must also know *how many* notes have been played on the instrument *prior* to the time at which the program has been joined.

Content that makes use of instruments that *preserve state* in this manner cannot be arbitrarily joined in-progress. If only note-on, note-off information is provided in the control bitstream, it is not possible to determine the correct state in which to begin at a midway point. To allow random tune-in, the author must include auxiliary information which periodically reconfigures the internal state (like a I-frame in a coded video sequence), to ensure that the correct synthesis is performed, or else use simpler algorithms with no state.

Facilities are provided in MPEG-4 for marking particular points in the bitstream as "break-in" points at which new joins are allowed. However, in Structured Audio streams, such points are not required, and in general are impossible to determine automatically. The musician or sound designer creating the bitstream must be aware of these considerations in authoring bitstreams if the resulting content is to be conveyed over channels which allow join-in-progress functionality. If the break-in points are not marked, the content cannot be correctly joined in progress.

3.3.2 Complexity considerations

As with bandwidth, the complexity analysis of algorithmic-synthesis bitstreams is difficult to pursue. The simplest sounds require less computation to decode with algorithmic synthesis than with wavetable synthesis; a sound like a pure sine, square, triangle, or other periodic waveshape requires only one table-lookup per sample, plus one increment of a phase pointer, to compute. In fact, simple algorithms in SAOL are the simplest of all MPEG-4 bitstreams to decode.

However, more complex algorithms may be readily transmitted as well. Since SAOL is a complete language, any algorithm at all may be downloaded and used for synthesis, and this includes algorithms which require arbitrarily large resources to decode. Further, unlike other codecs in MPEG-4 (including Wavetable Profile Structured Audio), the resources required to decode the bitstream cannot be determined from a simple examination. Decoding, simulated decoding, or equivalent processing is required to precisely determine the amount of memory, number of operations, and so forth necessary for decoding.

Several tools are available for instrumenting and analyzing a bitstream in decoding or simulated decoding, and are provided as informative additions to MPEG-4. The levels for supported complexity within the Full Profile are not yet completed, but will make use of a "feature vector" that includes elements such as RAM usage, instructions/sample, and test and branch instructions. Each level will require real-time decoding of bitstreams with less than a certain value for each of these elements, measured using one of the supplied instrumentation tools.

The implementation of Full Profile Structured Audio requires a programmable system, whether software-only on a host processor, or making use of an embedded, programmable DSP accelerator. At the current time, such technology is slightly beyond the state-of-the-art in audio decoding systems. Recently, Analog Devices has developed a real-time DSP accelerator for Csound processing [21], but this system is not embedded in a streaming context, and the Csound language is somewhat simpler than the MPEG-4 SAOL language. To develop and implement a conformant SAOL system requires significant expertise in audio transport, just-in-time compiler design, and real-time music synthesis – three areas of engineering study which normally have little interaction.

3.3.3 Advanced functionality

The primary advantage of the algorithmic-synthesis capability in MPEG-4 is flexibility. Particularly in interactive multimedia, SAOL and SASL offer sound designers more sonic capability than wavetable synthesis, even with downloaded samples. The broad scope of synthesis techniques, such as physical models, that allow more expressivity in performance than wavetable techniques are easily accessible through the MPEG-4 Structured Audio Full Profile. For use in videogames and virtual-reality environments, advanced "synthetic Foley" instruments [1] can provide highly-controllable parametric

sound effects. Again, such techniques are not available through wavetable synthesis, but require Structured Audio capability to implement.

As well as expressive sound synthesis, complex sound-generation algorithms may be written in SAOL and transmitted as content in Structured Audio. Rather than "scoring", or precisely scripting, all of the sound content, some of it may be created on-the-fly using algorithmic composition or virtual performance techniques. Conveying sound in this way is vastly more efficient than conveying it in a scored performance, since as much sound as desired may be created using essentially fixed bandwidth.

Finally, the advanced-effects component of AudioBIFS requires that a Full Profile Structured Audio decoder be present in the terminal. The capabilities required to implement the **AudioFX** node in AudioBIFS are the same as those required to provide audio-synthesis functionality in Structured Audio. Thus, applications which require more sophisticated client-side post-production than simple mixing – for example, parametric EQ, reverberation, or dynamic filtering – require full Structured Audio capability.

At the present time, taking advantage of Structured Audio capability requires significant knowledge of music-synthesis technology and algorithm development. Since the bitstream incorporates the synthesis algorithms, it is the content author rather than the standards body who is responsible for creating them and assuring that they generate the desired sound. As the standard matures in development, authoring tools which assist in this process by providing graphical front-ends will become available to assist the human author.

3.3.4 Applications for algorithmic synthesis

From the above analysis, we can arrive at a set of primary applications for SAOL-based algorithmic synthesis. Key application considerations for the use of SAOL include the following:

- Sophisticated content requiring high-quality synthesis
- Expressive content controlled interactively
- Sound design and synthesis expertise for bitstream creation
- If session tune-in capability is required, then less synthesis flexibility must suffice, or else local storage, existence of a back-channel, and/or high-bandwidth channel capability must be available.
- High-quality client-side post-production and effects.

Such applications today include videogames, low-bitrate Internet delivery of popular music, virtual-reality models and entertainment, psychoacoustic demonstrations, and contemporary electro-acoustic music. In the near future, applications such as interactive set-top boxes and collaborative networked music systems will also make use of MPEG-4 Structured Audio capability. The key capability of MPEG-4 in these contexts is the ability to precisely control sound synthesis and interactive client-side post-production.

4 HYBRID ALGORITHMIC/WAVETABLE SYNTHESIS

In this section, we provide speculative thoughts about interesting possibilities in hybrid algorithmic/wavetable synthesis. In the Full Profile of MPEG-4, the wavetable synthesis and algorithmic synthesis capabilities may be used together, where algorithms control the wavetable synthesizer (what notes it plays), and may be used to post-process the sound after it is created with wavetable methods. SAOL itself may also be used to deliver complete wavetable synthesizers, to generate sound from samples using other methods than those standardized as the SASBF semantics.

4.1 Algorithmic control of wavetable synthesis

In the Full Profile of MPEG-4, SASBF synthesis is not executed directly from incoming MIDI data as it is in the Wavetable Profile. Rather, the MIDI instructions are intepreted according to the general Structured Audio MIDI semantics – they instantiate SAOL-based synthesis rather than SASBF synthesis, the controllers have no direct effect but are exposed to the SAOL code, and so forth. To use SASBF synthesis

in this paradigm, a special SAOL command called **sbsynth** is provided. **sbsynth** performs "one note" of wavetable synthesis for given parameters and returns the audio result to SAOL for future processing.

Using this method, both simple manipulations and complex ones are possible. In the simplest case, the SAOL instrument receiving MIDI events passes them on without modification to the **sbsynth** instruction, and outputs the resulting sound. The synthesis performed in this case is the same as when the MIDI stream is processed by a SASBF-only synthesizer; the advantage is that SASL control of algorithmic synthesis is possible at the same time, and the resulting algorithmic and synthetic sounds are synchronized in playback.

More complex manipulation of the wavetable synthesis process is also possible. For example, rather than passing all events through to the SASBF synthesizer, the SAOL instrument receiving MIDI may filter some out, and only use the SASBF synthesis to generate sound for a particular range or subset of notes. The other notes are processed using a different instrument or ignored. Other possible manipulations include changing the notes to be synthesized (for example, inverting the keyboard) or algorithmically altering their velocity or duration.

The SAOL instruments may themselves generate the MIDI data which drives SASBF synthesis. For example, a SAOL instrument may generate multiple notes in response to an incoming note-on event, allowing complex layering of instruments. Or algorithmic composition tools written in SAOL may be controlled with SASL parametric controllers and generate **sbsynth** instructions to do the synthesis. The advantage of such a technique is that advanced flexibility of SASL-based control may be leveraged, while still utilizing the reduced complexity of SASBF for synthesis.

4.2 Algorithmic post-processing of wavetable synthesis

In the above examples, while the control of synthesis is carefully controlled in SAOL, the SASBF synthesis is the end-result and is played for the listener. However, this is not required. The sound produced by the SASBF synthesizer when it is controlled by the **sbsynth** command may be captured and processed further in SAOL.

SAOL has powerful capabilities for describing novel filters and transforms. Commands such as **fir**, **iir**, and **biquad** allow the precise delivery of time-varying filters which can be used to post-process wavetables. Other manipulations such as unusual chorusing and flanging are easily achieved using the **fracdelay** command, which provides a flexible multi-tap variable-position fractional delay line prototype. All of these capabilities may be executed on sounds generated with direct algorithmic synthesis, or on sounds generated with SASBF synthesis.

The SAOL primitives are also ideal for delivering sophisticated reverberation and equalization functions. Using hybrid algorithmic/wavetable synthesis, high-quality reverbs may be designed and delivered so that it is not necessarily to rely on the capabilities and sound quality provided by a particular manufacturer's hardware. An interface for accessing 3-D spatial audio is also provided; this allows sounds output from the algorithmic and/or wavetable synthesizers to be spatially positioned around the listener. Finally, by using the capabilities of the AudioBIFS **TermCap** node, the speaker configuration of the listening setup may be determined, and spatialization or auralization algorithms which have been appropriately customized may be downloaded and used for synthesis.

4.3 Downloadable wavetable synthesizers

If different capabilities for wavetable synthesis are desired than those directly provided by the SASBF part of the audio standard, new configurations may be easily downloaded. The capabilities of SAOL are well-suited to the customizable delivery of new architectures for wavetable synthesis. Sound samples may be transmitted at any desired sampling rate and then modulated to the correct frequency for playback; envelopes, LFO generators, ramps, and other common signal manipulators are all primitive elements of the SAOL language. Also, in an MPEG-4 terminal which provides natural-sound compression as well as sound-synthesis capabilities, the samples may be compressed in a variety of ways for transmission. For samples in which unusual manipulation is desired, high compression is often not possible, since the noise

added in the perceptual-compression process (which is inaudible under typical playback) will emerge under filtering and pitch-shifting operations. However, sample-compression can be highly effective for long-term, unshifted samples such as drum loops or other "one-shots".

5 CONCLUSIONS

We have presented a discussion of the capabilities, prerequisites, and applications of the wavetable and algorithmic synthesis tools in the MPEG-4 Structured Audio standard. MPEG-4 Structured Audio is the first international standard covering sound synthesis; as such, it enables a wide range of applications which are not possible with other standards, and broader buy-in than is possible with other general-purpose synthesis tools. As MPEG-4 Structured Audio decoders, content, and authoring tools become available, there will be a great move in the multimedia audio world towards the development of new types of audio content, and new capabilities in traditional content.

To thoroughly understand the capabilities of MPEG-4 Structured Audio is a more difficult task than with other methods for transport of synthetic sound. Musicians, sound designers, and developers who plan to make use of this new technology are encouraged to read the full Structured Audio specification, available via the WWW at http://sound.media.mit.edu/mpeg4. Other information, and a free (though non-real-time) implementation is also available at that site.

To conclude, it is important to remember that the MPEG process is ongoing. This paper does not represent a finished standard or the official views of ISO/IEC JTC/SC29/WG11, which is the supervisory body for MPEG, but the views of two experts involved in the standards development effort. Future improvements and clarifications to MPEG-4 before its completion in December 1998 may invalidate certain statements herein; suggestions for changes to the standard are welcome at any time.

6 ACKNOWLEDGEMENTS

The first author wishes to thank Prof. Barry Vercoe and the Machine Listening Group at the MIT Media Laboratory for helpful discussions, suggestions, and critiques relating to the present paper. The Structured Audio capabilities in MPEG-4 have had many contributors, including but not limited to Barry Vercoe, Michael Casey, Paris Smaragdis, Bill Gardner, Itaru Kaneko, Shigeki Fujii, Giorgio Zoia, Jyri Huopaniemi, Jens Spille, Brian Link, Luke Dahl, Tom White, Todor Fay, and Billy Brackenridge.

7 REFERENCES

[1] M. A. Casey, *Auditory Group Theory with Applications to Statistical Basis Methods for Structured Audio*, unpublished Ph.D. dissertation, MIT Media Laboratory (1998).

[2] M. A. Casey and P. Smaragdis, "NetSound", *Proc. Int. Computer Music Conf.*, Hong Kong (1996).

[3] R. Dannenberg, "Machine Tongues XIX: Nyquist, a language for composition and sound-synthesis," *Computer Music J.*, vol 21 no 3, pp. 50-60 (1997).

[4] W. G. Gardner, "Reverberation algorithms", in Brandenberg K. and M. Kahrs (eds*), Applications of Signal Processing to Audio and Acoustics* (Kluwer Academic Publishing, New York, 1998).

[5] A. Gersho, "Advances in speech and audio compression," *Proc. IEEE*, vol. 82, pp. 900-918 (1994 Jun).

[6] B. Grill, B. Edler, I. Kaneko, Y. Lee, M. Nishiguchi, E. Scheirer, and M. Väänäänen, eds, *ISO/IEC 14496-3 (MPEG-4 Audio) Final Committee Draft.* ISO/IEC JTC1/SC29/WG11 (MPEG) document N2203, Tokyo (1998).

[7] Headspace, Inc, "Rich Music Format", white paper available at http://beta.choicemall.com/thomasdolby/rmf-whitepaper.html (1996).

[8] J. Makhoul, "Linear prediction: a tutorial review", *Proc. IEEE*, vol. 63, pp.561-580 (1975 Mar).

[9] M. Mathews and J. Miller, *Music IV programmer's manual*. Murray Hill:Bell Telephone Laboratories (1963).

[10] J. McCartney, "SuperCollider: A new real-time sound synthesis language," *Proc. Int. Computer Music Conf.*, Hong Kong (1996).

[11] Microsoft Corp, "DirectMusic", white paper available at http://www.asia.microsoft.com/DirectX/pavilion/future/dmusic.htm (1998).

[12] MIDI Manufacturers Association, *The Complete MIDI 1.0 Detailed Specification v.96.2* (1996).

[13] MIDI Manufacturers Association, *MIDI Downloadable Sounds Level 1 Specification* (1998).

[14] S. Quackenbush, "Coding of natural audio in MPEG-4", *Proc. IEEE ICASSP*, Seattle, WA (1998).

[15] D. Rossum, "The SoundFont 2.0 file format," white paper available at http://www.soundfont.com (1995).

[16] E. D. Scheirer, "The MPEG-4 Structured Audio standard", *Proc. IEEE ICASSP*, Seattle, WA (1998).

[17] E. D. Scheirer, "SAOL: The MPEG-4 Structured Audio Orchestra Language", *Proc. Int. Comp. Music Conf.*, Ann Arbor, MI, 1998.

[18] E. D. Scheirer, "Structured Audio and effects processing in the MPEG-4 multimedia standard", *ACM Multimedia Sys. J.*, to appear.

[19] J. O. Smith, "Physical modeling using digital waveguides," *Computer Music J.*, vol. 16 no. 4, pp. 74-91 (1992).

[20] B. L. Vercoe, *Csound: A Manual for the Audio Processing System*, MIT Media Laboratory (1995).

[21] B. L. Vercoe, "Extended Csound", *Proc. Int. Computer Music Conf.*, Hong Kong (1996).

[22] B. L. Vercoe, W. G. Gardner, and E. D. Scheirer "Structured Audio: Creation, transmission, and rendering of parametric sound representations." *Proc. IEEE*, vol 86, pp. 922-940 (1998 May).