MPEG-4 Structured Audio Systems

Mihir Anandpara
The University of Texas at Austin
anandpar@ece.utexas.edu

Abstract

The MPEG-4 standard has been proposed to provide high quality audio and video content over the Internet. This content is represented in the form of audiovisual objects. However, different parts of the audiovisual scene are encoded seperately depending on the nature of the data to be encoded. The standard calls for aggresive coding techniques which ensure reception of high quality audio and video at low bit rates. One such encoding scheme for audio, generally applied to synthetic audio and sound effects, is known as Structured Audio. This paper presents a survey on the structured representation of sound and the ways it can be decoded and synthesized at the receiving terminal.
All the different components of the audio scene are decoded seperately (Structured Audio is one of the components). The different decoded streams are then composed into one coherent sound signal and presented to the user. This audio composition in MPEG-4 is accomplished by the AudioBIFS layer in the MPEG-4 decoder. The nodes in this layer compose the different sounds together and also provide some abstract effects and virtual reality effects post-processing on the entire audio scene. Some of the post-processing can also be represented in the structured audio format that is used for synthetic sound encoding. This project focuses on the working of the AudioBIFS layer, specifically the nodes in this layer which rely use a structured representation encode postprocessing of audio effects, such as reverbrations.

I. Introduction

Streaming audio and video content on the Internet has become increasingly popular. Several standards have been proposed for dealing with streaming audio and video. MPEG-4 is the first standard that addresses presentation content as a set of audio visual objects. The main functionalities in MPEG-4 are content-based coding, universal accessibility, and good coding efficiency [2]. Using the new MPEG standard will result in audio, speech and video representations which can be transmitted at very low bit rates, but render high fidelity [11].

Traditional audio coding techniques can be divided into two categories. Lossless encoders remove entropic redundancy from the sound signal. This redundancy exists due to the fact that successive samples of the data are correlated, and some redundancy may be eliminated using this principle. Also, more frequently occuring samples can be encoded with shorter codes than less frequently occuring samples, as in Huffman coding. On the other hand, lossy encoders (MP-3, Real Audio) remove perceptual redundancy from a sound signal. These encoding schemes remove those details

from the sound signal that cannot be perceived by the human ear. They use physo-acoustic principles to determine what part of the signal is redundant and can be eliminated.

The MPEG-4 standard has been developed for state-of-the-art representation, transmission and decoding of multimedia objects at a low bit-rate. MPEG-4 audio can be used for every application requiring the use of advanced sound compression, synthesis, manipulation and playback [3]. The traditional coding techniques discussed above are not enough to represent audio signals containing a large amount of musical content or sound effects, and still maintain bandwidth efficiency. However, sound signals, especially music signals, represent another level of redundancy known as *structural redundancy*. In a soundtrack, many notes or sound events sound the same or very similar. Also, many soundtracks contain repeating patterns, such as drumbeats. If all parts of the soundtrack can be represented symbolically, a lot of redundant information can be eliminated [5]. This characteristic of soundtracks motivates the use of a symbolic and structured technique for representing sound signals which can be transmitted at a much lower bandwidth. This symbolic representation of sound is referred to as structured audio in MPEG-4.

Structured audio transmits sound in two parts: a description of a model and a set of parameters that make use of the model [1]. Every different signal or sound in a soundtrack is mapped to a particular model. Each model has several parameters associated with it. The parameters can be interpreted as high level features about the sound signal. They can be varied to control various perceptual aspects like amplitude and pitch. This kind of structure and flexibility lends itself well to applications involving synthesized sound, since the sound of each instrument can be associated with a specific model and decoded accordingly at the reciever.

This paper is organized as follows. Section 2 gives a detailed description of the stuctured audio component of the MPEG-4 standard, including encoding, parameter representation techniques and

audio postprocessing. Section 3 deals with decoding and synthesis of sound from a structured audio representation. Section 4 gives an overview on the objectives and goals of this project.

II. Structured Audio in MPEG-4

A. Representation of structured audio in MPEG-4: SAOL

The MPEG-4 Audio standard (ISO/IEC 14496 - Part 3) [3] consists of a subsection on representation, decoding and synthesis of structured sound. There are many ways to obtain and use a structured representation of signal, described in greater detail in [5]. The MPEG-4 standard allows for sound-synthesis algorithms to be specified as a computer program. Several computer music languages [9], [4] have been developed, which use the concept of *unit-generators*. Unit generators are simply functional blocks like oscillators, filters and envelopes, that are connected together to form a network for the signal path flow.

A new language known as Structured Audio Orchestra Language (SAOL) has been developed for representation of structured audio and effects in MPEG-4 audio scenes. Any MPEG-4 structured audio scene can be divided into two parts - the *orchestra* and the *score*. SAOL defines an orchestra as a set of *instruments*, where each instrument corresponds to a seperate model, and describles some digital signal processing algorithm that produces or manipulates sound. The score contains information which can be used to control parameters to the signal processing algorithms described in the orchestra at run-time. The structured audio score is encoded in another language, known as the Structured Audio Score Language (SASL). This seperation of the algorithm description and control lends flexibility and modularity in the design of sound. In order to ensure backward compatibility with other music synthesis systems, the standard also allows for control of SAOL code through MIDI as well as wavetable synthesis [6].

B. Audio decoding in MPEG-4

Sound in the MPEG-4 bitstream is coded in several different ways, depending upon the coding efficiency desired and the nature of the sound signal that has to be encoded. For example, simple music can be coded using a *General Audio* coder based on perceptual (or natural audio) coding techniques described in the previous section. This encoder generally produces a bitstream in the range of 16-64 kbps [12], [10]. More complex soundtracks with synthetic components and effects may be coded in SAOL, and speech may be coded using a speech coder. All the seperately coded streams are multiplexed together and transmitted.

At the decoder terminal, these multiple multiplexed bitstreams are demultiplexed and each bitstream is decoded seperately using the corresponding decoders. There are seperate decoders for speech, natural audio, and synthetic audio. The output of the different decoders are uncompressed *primitive media objects*.

The structured audio part of the bitstream *header* that is transmitted at the beginning of the decoding session contains the orchestra description, and a MIDI/SASL score file. The decoding terminal consists of a *scheduler*, which compiles the orchestra for use by a *signal processing* system, which executes algorithms written in SAOL. The scheduler also reads the control data from the score and changes the parameters of the signal processing system in order to change the output at run-time. The output of this decoding process is an uncompressed primtive media object.

C. AudioBIFS: Sound Composition and Effects Processing

The primitive media objects output by the different decoders are not played directly. Instead, these different objects are composited together into one coherent audio signal and presented to the user. The processing layer which accomplishes this task is known as Audio Binary Information for Scene Descripition (AudioBIFS), which is a part of the BIFS standard defined for compositing

the entire MPEG-4 scene from different audio and video objects and presenting it to the user. The AudioBIFS standard is described in detail in [3],[7]. The AudioBIFS layer also provides effects postprocessing, such as reverbrations or spatialization on the composited sound scene. Sound can be postprocessed with arbitrary downloadable filters, reverbrators, and other digital audio effects [7].

The AudioBIFS layer uses *scene graph* concepts to organize and compose audio material. Any node in the graph may describe any object or sound signal, or some parameters of the signal, or some transformations on the signal. One of the transformation nodes defined in the AudioBIFS standard is the *AudioFX* node. This node allows for custom signal processing effects to be applied to its input signal. These effects are described in SAOL. The AudioFX node gives the sound designer/composer some flexibility into designing custom effects for each application, since they are synthesized at run-time and depending upon the effects described in the SAOL program. Just as in the case of structured audio synthesis, time-varying parametric effects can also be incorporated at the AudioFX node through an SASL score. A detailed description of the AudioFX node and other nodes in the AudioBIFS scene graph is presented in [7].

III. Synthesis of Structured Audio

The main complexity in the implementation of the MPEG-4 structured audio system is in the decoder/synthesizer. The first step in the synthesis of audio from a structured audio representation is to compile the SAOL program and interpret the different operations that are indicated as a part of the program. Then, scheduler in the SA decoder must control the signal processing system to produce output in real-time. An interesting feature of SAOL is that variables in any instrument, on which the signal processing is defined can be modified at two rates, the *control rate* (krate) and the *audio rate* or sampling rate (srate). The control rate variables are the variables which represent the parameters of the model which can be modified in the score. The audio rate variables have to be updated at the sampling rate. They represent the sound signal output by any instrument in the orchestra. Generally,

the control rate is lower than the audio rate. This rate distinction provides the basis for defining blocks, where the size of each block is the ratio of the audio rate to the control rate. All processing of audio rate variables for one value of every control rate variable can be done as a block, instead of being done on a sample-by-sample basis. This can lead to efficient block based implementations of the synthesis algorithms, which can be implemented on vector-processors or SIMD/VLIW DSPs or multimedia processors.

One of the proposals for implementing the structured audio decoding and processing system is to use an interpreted language approach like Java. The SAINT (Structured Audio Interpreter) [8] system produces *macroinstructions* for a *virtual DSP* which has been designed such that arithmetic and logic operations can be defined for vectors of values, and is based on a vectorial instruction set. This setup is therefore divided into two layers: a platform-independent layer, which produces the virtual DSP macroinstructions, and a platform-dependent layer, which converts the macroinstructions into instructions optimized for the particular processor on which it is running.

The compiler for the SAINT system has the ability to recognize and mark out parts of the instructions that can be executed in blocks. Processing of the signal can generally be done in blocks unless there is some inherent feedback in the signal processing operation. One of the most common examples of feedback is an IIR filter operation defined on audio rate variables in the SAOL orchestra. Such instructions have to be executed on a sample-by-sample basis. All other instructions can be executed on a block-by-block basis. This use of data-level parallelism can greatly reduce the time required to synthesize the audio. Simulation of this system shows that it gives a much faster decoded output than the software compilers provided as an MPEG-4 reference software for SAOL, known as saolc [1].

IV. PROJECT GOALS

The main goal of this project is to design and simulate the AudioBIFS system for composition of audio scenes in MPEG-4. This will involve simulation of all the nodes on the AudioBIFS scene graph as well as the interaction between the different nodes. This simulation can be done using either with a Synchronous Dataflow Model (SDF) or a Process Network (PN) model. The advantage of the SDF model is that the schedules are determined at compile time, and it lends itself very well to signal processing applications consisting of several processing blocks performing different functions. The advantage of the PN model is that we can execute some nodes in parallel and may yield higher utilization and throughput for the system. This parallel execution can also be used to figure out what parts of the AudioBIFS system can be run in parallel without changing the output of the system.

Another goal of this project is to study the effects processing algorithms described in a structured representation, and extract parallelism at the data and instruction level, which would enable more block based execution of the algorithms on high performance DSPs or multimedia processors. This investigation will be on the lines of the analysis of SAOL programs in [8]. I intend to look at ways in which the synthesis process can be further improved by running different parts of the algorithms in concurrent threads. This idea can then be expanded to work on multi-processor systems and distributed systems to provide faster execution of algorithms and synthesis of sound.

REFERENCES

[1] "MPEG-4 structured audio." [Online]. Available: http://web.media.mit.edu/ eds/mpeg4/sa-tech.html

[2] A.Puri and A.Eleftheriadis, "MPEG-4: An object-based multimedia coding standard supporting mobile applications." [Online]. Available: citeseer.nj.nec.com/puri03mpeg.html

[3] I. Y. M. B.Grill, B.Edler and E.Scheirer, "ISO/IEC JTC1/SC29/WG11 (MPEG) document N2203," in *ISO/IEC* $11496-3$ *(MPEG-*4 *Audio) Final Committee Draft*, 1998.

[4] B.L.Vercoe, *CSound: A manual for the audio processing system*, Massachusetts Institute of Technology Media Lab, 1996.

[5] W. B.L.Vercoe 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, 1998.

[6] E.D.Scheirer, "Algorithmic and wavetable synthesis in the mpeg-4 multimedia standard," in *Proceedings of the* 105^{th} *Convention of the Audio Engineering Society (AES)*, 1998.

[7] J. E.D.Scheirer, R.Vaananen, "AudioBIFS: Describing audio scenes with the MPEG-4 multimedia standard," *IEEE Transactions on Multimedia*, vol. 1, no. 3, pp. 237–250, 1999.

[8] G.Zoia and C.Alberti, "A virtual DSP architecture for audio applications from a complexity analysis of MPEG-4 structured audio," *IEEE Transactions on Multimedia*, vol. 5, no. 3, pp. 317–328, 2003.

[9] M.Matthews, *The Technology of Computer Music*. MIT Press, 1969.

[10] N. N.Jayant and R.Safranek, “Signal compression based on models of human perception,” *Proceedings of the IEEE*, vol. 81, pp. 1385–1422, 1993.
[11] E. Scheirer, “Structured audio and effects processing in the MPEG-4 multimedia standard,” *Multimedia Systems*, vol. 7, no. 1, pp. 11–22, 1998.
[12] S.R.Quackenbush, “Coding of natural audio in MPEG-4,” *Proceedings of IEEE Int. Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3797–3800, 1997.