Proceedings, IEEE Int'l Symposium on Circuits and Systems, Hong Kong, June 1997 (invited paper).

The MPEG-4 System and Description Languages: From Practice To Theory

Alexandros Eleftheriadis

Department of Electrical Engineering
Columbia University
New York, NY 10027, USA
eleft@ee.columbia.edu

ABSTRACT

We provide an overview of the novel system architecture currently adopted in the ISO MPEG-4 standardization effort, as embodied in its System and Description Languages design. We briefly describe its composition capabilities, multiplexing and sychronization model, as well as the syntactic description mechanism which is based on our "Flavor" language. We also describe a theoretical framework for the analysis of algorithmic or language-based compression using "Complexity Distortion Theory," an extension of Kolmogorov Complexity theory that takes into account distortion, and show that it is a universal and the most natural framework for analyzing traditional and new compression techniques.

1. INTRODUCTION

For the past several decades, digital audio-visual information representation has been focused on compression. Existing standards have been very successful in addressing particular application needs, such as H.261 and H.263 for video conferencing, MPEG-1 for CD-ROM type video, and MPEG-2 for broadcast TV. The diversity of current and foreseeable multimedia applications, however, makes it very difficult to satisfy their requirements with such monolithic vertical designs. MPEG-2 attempted to accommodate this need for flexibility by defining several profiles and levels, each one addressing particular application needs.

In addition, the domain of audio-visual information is increasingly encompassing non-traditional (or non waveform-based) areas, including synthetic-natural hybrid representation and even purely synthetic material (graphics, MIDI). We describe this trend using the term "algorithmic representation," in the sense that the content may: 1) include downloadable algorithms that process it, and 2) include downloadable or fixed algorithms that generate it. The latter case includes computer graphics and synthetic audio, but is also applicable to natural material (e.g., in model-based video coding and others [1]).

The on-going MPEG-4 standardization effort is intended to offer more flexibility and extensibility than traditional approaches by using the so-called MPEG-4 System and Description Languages (MSDL) framework. This framework is the mechanism with which all MPEG-4 components are described and configured, and represents a radical departure from past MPEG designs. It allows MPEG-4 to address exciting new capabilities such as object-based coding and programmability using platform-independent downloadable code.

We briefly describe the various MSDL components, including composition, multiplexing and synchronization, and the syntactic description capabilities.

We also show that this new generation of representation techniques, which includes programmability and synthetic or algorithmic material, requires new mathematical tools to properly address questions of efficiency and performance. We outline a new theoretical framework that we are developing, called "Complexity Distortion Theory," which extends traditional Kolmogorov Complexity theory to account for distortion.

We should note that the MPEG-4 MSDL specification (currently described in its Working Draft [2] and Verification Model [3]) is still evolving; the final standard is scheduled to be ratified in November 1998.

2. THE MPEG-4 ARCHITECTURE

The MPEG-4 architecture considers a system for communication of audio-visual information in the form of objects. These objects are compressed, error protected (if needed) and multiplexed at the encoding side, and subsequently demultiplexed, error-corrected, decompressed, and composited at the receiving side. This architecture includes the traditional components of natural audio-video compression and multiplexing, but it also adds several new ones:

- composition, occurring at the receiving terminal prior to presentation,
- representation of synthetic media,

- flexibility (configuration or extension of an MPEG-4 system).

A fundamental component of this new architecture is the notion of objects. MPEG-4 objects are objects in the traditional object oriented sense, i.e., entities that combine a data structure (defining the object's state) with a set of methods (defining the object's behavior). A method is an executable procedure associated with an object that operates on information in the object's data structure. Classes are templates for objects. MPEG-4 will standardize a number of pre-defined classes, forming the MPEG-4 standard class library.

In order to allow for flexibility, two categories of profiles are envisioned: flexible, and non-flexible. In flexible profiles, the user or encoder will be able to produce new encoder-defined classes, modify the bitstream syntax of existing classes, and potentially download new methods. In a non-flexible profile, configuration can be provided via pre-defined (standardized) option selectors that must be present in the bitstream, as in MPEG-2.

Figure 1 shows the overall structure of a (flexible) MPEG-4 terminal with the various content processing steps. Considering a decoder's operation, the data is

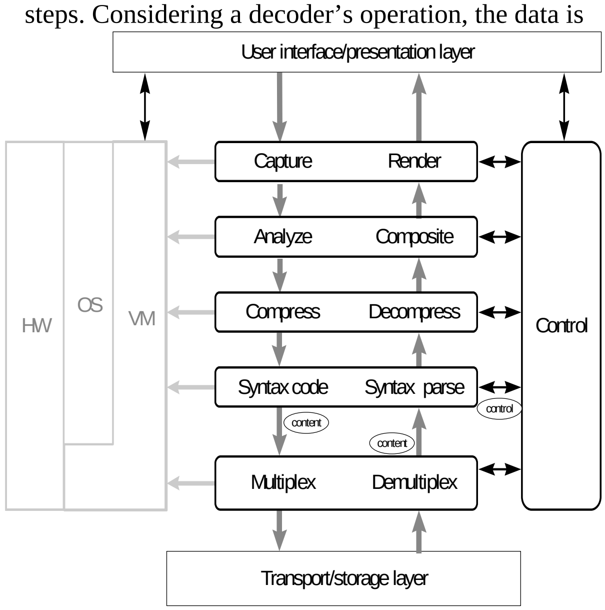

Figure 1: MPEG-4 terminal structure.

retrieved from the network or storage and then demultiplexed into elementary objects. These objects are then parsed from the bitstream and decoded using their corresponding decoding techniques. Finally, a set of such objects are composited together and rendered into the final product which is presented to the user.

The hardware (HW) and Operating System (OS) components are implementation-dependent; the Virtual Machine (VM) component, however, is implementation-independent and provides for platform-independent downloadable code support. Java [9] has been currently selected to implement the flexible portion of the control component of the terminal, and hence the Java VM is used.

The "control" box, together with its interfaces to the various content processing components, is the domain of MSDL. It includes an executive, a compositor, and a presenter. The executive is responsible for the overall execution of the object's methods, while the compositor coordinates the aggregation of object information that will constitute the finally presented imagery or audio. The presenter deals with user interaction.

Each audio-visual object class includes its own render, decode, parse, and handle methods. The parse method is responsible for interpreting the bitstream syntax and initializing the object's state. The decode method uses this information to perform traditional decompression of the object at hand. The parse and decode methods are together part of individual classes that are programmatically attached to objects, hence allowing an object to use different decoders as needed. The render method is responsible for the generation of the corresponding pixels/samples according to information provided by the compositor, which are subsequently used by the compositor to formulate the final image displayed or sound played. The handle method is responsible for reacting to user-generated events (user interaction).

In order to allow reuse of objects in the construction of new classes, flexible MPEG-4 terminals need to provide a standardized API (Application Programming Interface) for accessing individual objects and their methods. This API is still under development, and will be exercised using the programming facility provided by MSDL (currently Java) in order to define new tools or algorithms.

3. COMPOSITION

In order to produce the final image or sound presented to a user, the spatio-temporal location of individual objects within a scene must be determined. This is accomplished using composition information that is transmitted together with the content. Depending on the application, the individual objects may be 2-D or 3-D; in all cases, they include a temporal component as well.

Each object has a local coordinate system, in which it has a fixed location and scale. Objects are positioned in a scene by specifying a coordinate transformation from the object's local coordinate system into a common, global coordinate system called the scene coordinate system. The transformation is not part of the object, but part of the scene itself; as a result, scene description is sent as a separate stream within the MPEG-4 multiplex. This separation facilitates bitstream editability and is one of the content-based functionalities of MPEG-4. Scene descriptions are hierarchical, forming a tree with the scene as the root object and more detailed objects as one traverses the tree towards its leafs.

In the case of 2-D non-flexible scene description, a specific bitstream syntax is defined for communicating the desired transformation. The transformation itself can be global translation, zoom, affine, or perspective. For flexible 2-D composition, the scene description is transmitted as a class.

For 3-D scene description a specific mechanism has yet to be adopted. Current activities focus in the use of VRML [8] as a scene description language. In particular, two different strategies are examined: use of VRML 2.0 extension capabilities to accommodate MPEG-4 objects, and extension of VRML 2.0 with MPEG-4 specific nodes.

4. MULTIPLEXING/SYNCHRONIZATION

Multiplexing provides facilities for deinterleaving audio-visual objects from a single bitstream, adaptation of timing information (clock recovery), synchronization of individual objects in time, and buffer control.

The buffer model assumes a constant delay from the multiplexer output to the demultiplexer input, and utilizes per-object decoding and presentation buffers. All objects have explicit or implicit presentation and decoding timing information. A global system time base is assumed at the decoder, whereas individual object time bases are allowed for different objects. In addition, object expiration time stamps are also considered, which indicate the time at which an object can be discarded from the decoder (decoded objects can thus be reused if needed). Object time bases are transformed to the system time base to ensure proper coordination of the presentation and decoding processes.

The multiplexer is constructed in a two layer configuration, and originates from the ITU-T H.223 Annex A design. The upper layer provides facilities for flexibly interleaving logical channels with widely varying instantaneous bit rate. As in H.223, it allows several logical channels to coexist in the same PDU. The logical channel configuration is communicated using signaling. The lower layer is responsible for providing robustness and quality of service support, and can be ignored if appropriate facilities are provided by the underlying network (e.g., it can be substituted by ATM, H.223., H.223A, TCP/UDP, or MPEG-2 TS). Facilities provided include header recovery, CRC and BCH coding. convolutional interleaving, as well as ARQ.

5. SYNTACTIC DESCRIPTION

MPEG-4 adopted the separation of bitstream parsing from the remaining decoding operations. Bitstream parsing (or syntax decoding) refers to the mapping of data from their bitstream representation to their native form for the particular architecture. A simple example is the mapping from a 3-bit integer in the bitstream to the native sign-extended representation in the terminal's architecture (e.g., 32 bits). The motivation is that one can change the bitstream representation without modifying the decoding algorithm in any way. This allows for significant flexibility in reconfiguring the bitstream syntax without requiring modification of existing decoding tools.

In order to describe the bitstream syntax a Syntactic Description Language (SDL) has been adopted, based on our Flavor design [5, 6]. SDL provides an extension of the typing system of regular object oriented programming languages like C++ and Java to include bitstream representation information. Use of a formal language to describe the bitstream syntax has several benefits. As described above, it allows for reconfiguration of the syntax without modifying the decoding tools. It allows for both forward and backward compatibility: if a new syntactic component is defined, older systems will simply discard them without using them whereas the tools of new systems will be able to use them. It provides for automatic generation of bitstream compliance testing and generation tools. Finally, it can be used to automatically generate the skeleton of an encoder/decoder, thus significantly simplifying the task of the developer of a software/hardware implementation tools.

Flavor/SDL is organized around the notion of classes, and provides support for a wide array of syntactic constructs. The example in Figure 2 illustrates the approach. We define a class called 'Example' con-

```
// Example class
class Example {
     int(3) a;
     int(a) b;
     if (b<=0) {
            int(c_map) c;
     } else {
            int(b) c;
     }
}

// Table used in Example class
map c_map(int) {
     0b0,   5,
     0b10,  10,
     0b11,  15
}
```

Figure 2: An example of Flavor/SDL.

taining the following elements: an integer 'a' that is represented using 3 bits in the bitstream (by default, most significant bit first), and an immediately following integer that is represented using 'a' bits in the bitstream. Then, if the value of 'b' is negative or zero, the integer 'c' is contained using the variable length code table 'c_map'; otherwise, 'c' has a fixed length of 'b' bits. The map defines that the bitstring '0' correspond to the value 5, '10' corresponds to the value 10, and '11' corresponds to the value 15.

As we can see, the bitstream representation information is embedded in the class definition, and hence all object information (data, syntax, methods) is located in one place. Note that the flow control commands ('if') is part of the class' declaration. More detailed information can be found in [2, 5]. A translator from Flavor to C++/Java has already been developed.

6. AN ALGORITHMIC REPRESENTATION THEORY

The MPEG-4 architecture clearly indicates that the notions of programmability and extensibility will soon find their way into mainstream decoding systems, and provide the means towards achieving a new level of integration in audio-visual information processing. Interestingly enough, however, our theoretical arsenal is not well equipped from a conceptual standpoint to address these new systems.

The traditional (waveform-based) representation theory is based on information and rate distortion theories t that were developed several decades ago. Several modern techniques, including fractals and model-based coding do not directly fit in the old framework. We have proposed [7] a new theoretical framework, called Complexity Distortion Theory, which attempts to provide the proper conceptual framework to analyze programmability and algorithmic content.

Our approach is based on the notion of a decoder as a programmable terminal. This means that the decoder is a Turing machine and hence that, in addition to data, it can receive algorithms that modify its operation. The question of lossless representation of an object by an algorithm has already been addressed by the work of Kolmogorov, Chaitin, and Solomonoff, who independently formulated the foundations of Kolmogorov Complexity Theory [10]. The complexity of an object 'x' is the length of the shortest program which, when run on a universal Turing machine, will output 'x'. It has been shown that complexity is assymptotically equivalent to entropy for stochastic ergodic sources.

We have extended Kolmogorov Complexity to include distortion, thus providing the means to address lossy audio-visual representation techniques. In [7], we show that, the same way Kolmogorov Complexity parallels entropy, Complexity Distortion parallels rate distortion by being assymptotically equivalent for stochastic ergodic sources. This essentially closes the circle between stochastic waveform-based theories, and the deterministic programmable complexity theories.

It is of course natural to expect that the different theories would predict identical bounds. Their difference, however, is in the conceptual tools that they provide to address important questions of efficiency and design in real systems. In particular, complexity theory is fully deterministic, and hence in addition to addressing programmability it also does not require knowledge of the statistical properties of the source. We are currently developing techniques to address questions of performance for real decoders with finite resources in terms of both space and time.

REFERENCES

[1] L. Torres and M. Kunt, eds., "Video Coding: The Second Generation Approach," Kluwer Academic, Boston, Massachusetts, 1996.

[2] ISO/IEC JTC1/SC29/WG11 N1483, MPEG-4 Systems Working Draft Ver. 2.0, November 1996.

[3] ISO/IEC JTC1/SC29/WG11 N1484, MPEG-4 Systems Verification Model Ver 2.0, November 1996.

[4] O. Avaro, P. Chou, A. Eleftheriadis, C. Herpel, and C. Reader, "The MPEG-4 System and Description Languages," *Signal Processing: Image Communiation*, Special Issue on MPEG-4, to appear in 1997.

[5] Y. Fang and A. Eleftheriadis, "A Syntactic Framework for Bitstream-Level Representation of Audio-Visual Objects," *Proc., IEEE Int'l Conf. on Image Proc.*, Lausanne, Switzerland, September 1996.

[6] A. Eleftheriadis, "A Syntactic Description Language for MPEG-4," Contribution ISO-IEC/JTC1/SC29/WG11 MPEG95/546, Dallas, Texas, November 1995.

[7] D. Sow and A. Eleftheriadis, "Complexity Distortion Theory," submitted to the *1997 IEEE Int'l Symp. on Inf. Theory and its Applications*, October 1996.

[8] A. L. Ames, D. R. Nadeau, and J. L. Moreland, "The VRML Sourcebook," Wiley, New York, 1996

[9] J. Gosling, B. Joy, and G. Steele, "The Java Language Specification," Addison Wesley, Reading, Massachusetts, 1996.

1.1. Li and P. Vitanyi, "An Introduction to Kolmogorov Complexity and its Applications," Springer Verlag, New York, 1993.