

Chapter 3

THE MPEG-4 STANDARD FOR INTERNET-BASED MULTIMEDIA APPLICATIONS

Charles Law and Borko Furht

Abstract

With the development of the MPEG-4 standard in 1998, a new way of creating and interacting with audio-visual media was conceived. This chapter provides an overview of this new standard. Topics covered in the chapter include a description of visual and audio objects, both natural and computer generated, and a description of how scenes are put together in MPEG-4. We also present an overview of the systems necessary for MPEG-4 operation and insight into possible Internet-based multimedia applications. We briefly discuss some research issues in MPEG-2 design and implementations.

1. INTRODUCTION

In the last several years, a number of standards for representation and coding of audio-visual data have been proposed and adopted. Video coding standards, their format, compressed bit rates, and applications are summarized in Table 1, adapted from [13].

The Moving Picture Experts Group (MPEG) was established in January 1988 with the mandate to develop standards for coded representation of moving pictures, audio, and their combination. The first standard, MPEG-1, specified coding of audio-visual data for bit rates of 1.5 Mbps, while the second standard MPEG-2 is more generic standard intended for a variety of audio-video applications at the bit rate in range of 3-40 Mbps.

A new generation of multimedia and video applications requires new functionalities that are not supported by previous MPEG-1 and MPEG-2 standards. Therefore, a request for proposal concerning the MPEG-4 standard was made in July 1995. This new standard (MPEG-4) is designed to provide an object-oriented and content-based methodology for producing, delivering, and consuming audio-visual content. MPEG-4 was finalized in October 1998. Currently, MPEG is working on a fully backward compatible extension titled "MPEG-4 Version 2." This extension should be finalized by January 2000.

Table 1. Video Coding Standards and Their Characteristics.

STANDARD	FORMATS	COMPRESSED BIT RATE	APPLICATIONS
H.261	QCIF CIF	p x 64 Kbps p=1,2,...30	Videophone and videoconferencing via ISDN.
H.263	SQCIF 16CIF	Flexible	From low bit rate videophone to high quality videoconferencing.
MPEG-1	SIF	1.5 Mbps	Interactive multimedia, storage and retrieval of VCR quality video on CD-ROM.
MPEG-2	Flexible	3-40 Mbps	Digital TV, HDTV, broadcasting satellite services, electronic cinema, home television theatre.
MPEG-4	Flexible	Flexible	Interactive multimedia, mobile multimedia, content-based storage and retrieval, video games, video broadcast, collaborative scene visualization.

MPEG-4 combines elements from three fields: digital television, interactive graphics, and the World Wide Web [14]. Unlike previous MPEG standards, MPEG-4 allows for separately coding and describing each content-based object comprising a scene. This means that objects such as background music, foreground speech, a graphically generated globe, a specific video object (such as a person), and scene background, can be individually defined and manipulated. In previous standards, the scene would have to be created as a singular audio-visual entity such as an MPEG-2 clip. Furthermore, because each item is separate, it can be individually removed from the scene. This means that applications, which are unable to compose an entire MPEG-4 scene, may scale the scene down to only critical aspects.

Additional functionality provided by MPEG-4 includes:

- Intellectual property rights management.
- Synchronization of audio-visual objects.
- User interaction with audio-visual objects (change viewing coordinates, enlarge or shrink an object, modify pitch, select object enhancements if offered, replace scene, etc.)
- Scaling of audio-visual objects.
- Error robustness.
- Transparent delivery of information either locally or remotely.

MPEG-4 provides some powerful tools in which to create scenes. To control composition of these scenes, MPEG-4 uses a binary language to specify scene description rather than imbed the information into the compressed objects. This makes MPEG-4 objects highly reusable. All of this gives the user much more control over the presentation of a scene than was available with the earlier standards. Figure 1, which illustrates typical MPEG-4 scene composition, shows several objects (desk, globe, blackboard, instructor, and audio) placed into a 3-D coordinate system relative to the supposed viewer [4].

1.1 OVERVIEW OF MPEG-4 VIDEO CODEC

A general architecture of MPEG-4 video codec (encoder and decoder), shown in Figure 2, is based on the concept of video objects [13,16]. Both video encoder and video decoder are composed of a number of video object encoders and decoders, which apply the same coding

scheme to each video object separately. The user can interact with the objects at the encoder or decoder side – operations include scaling, dragging, and linking objects.

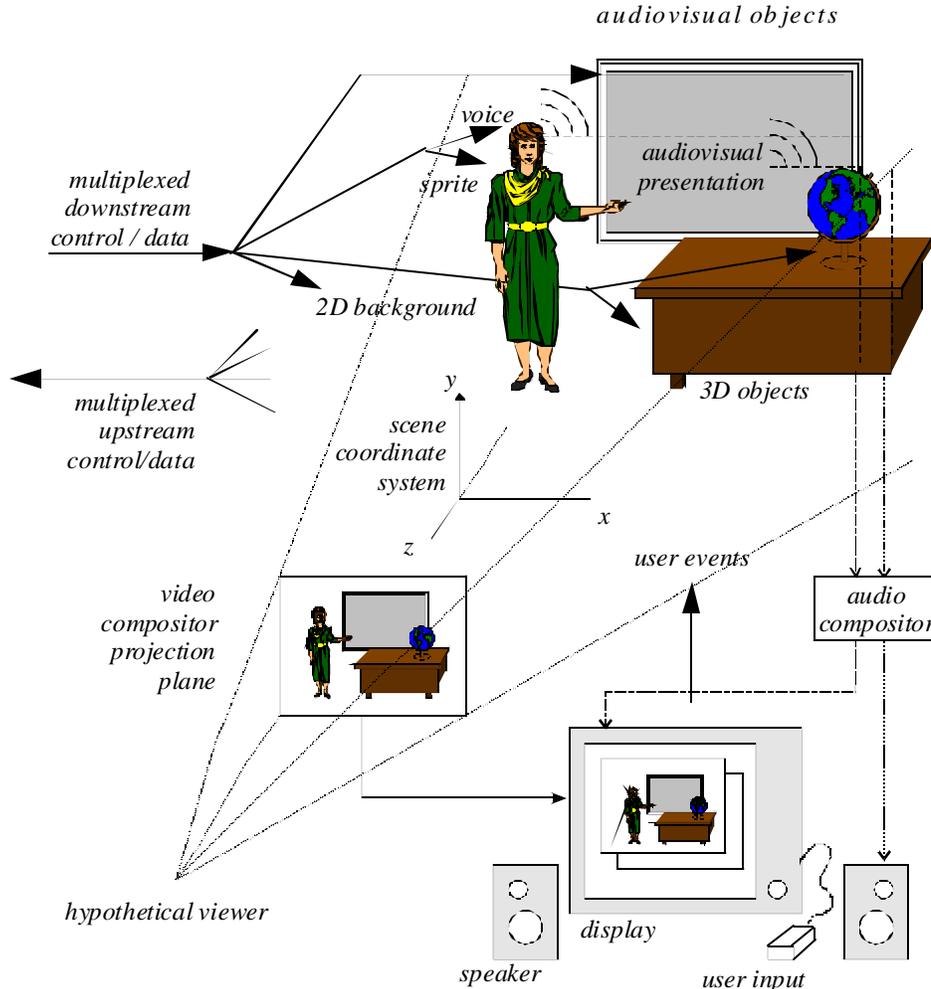


Figure 1. MPEG-4 scene (from [4]).

Figure 3, adopted from [13], illustrates an example of video coding and composition using MPEG-4. Several objects, such as *Man*, *Car*, and *House*, are separated from the original input video. Each video object is then encoded separately by video object (VO) encoder and then transmitted through the network. At the receiving side, these objects are decoded separately using VO decoder and sent to the compositor. The user can interact with the compositor to reconstruct the original scene (a), or to manipulate objects and create different scenes (b). In addition, the user can request from the compositor to download new objects from a local or remote database libraries, and insert or replace objects in the scene (c).

The main components of a MPEG-4 video coder include [13]:

- Shape coder,
- Motion estimation and composition, and
- Texture coder.

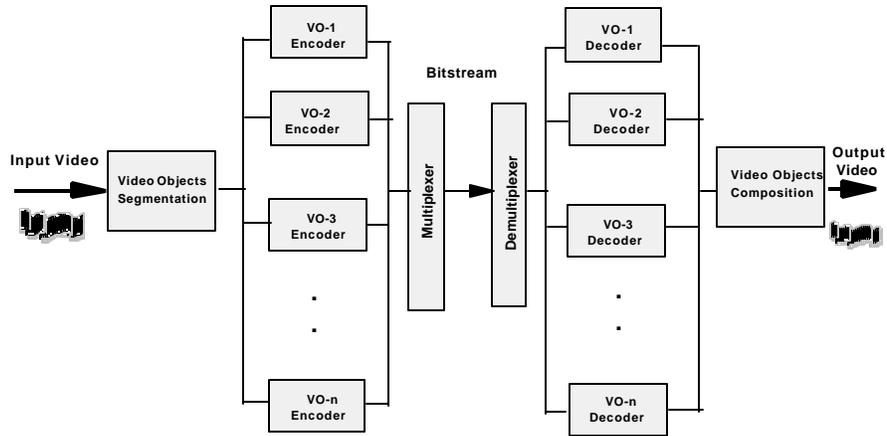


Figure 2. The architecture of MPEG-4 video codec.

The shape coder is used to compress the segmentation information, which specifies the object region and contour within the scene. Motion estimation and composition is used to reduce temporal redundancies. The texture coder is applied to intra and residual data after motion compensation.

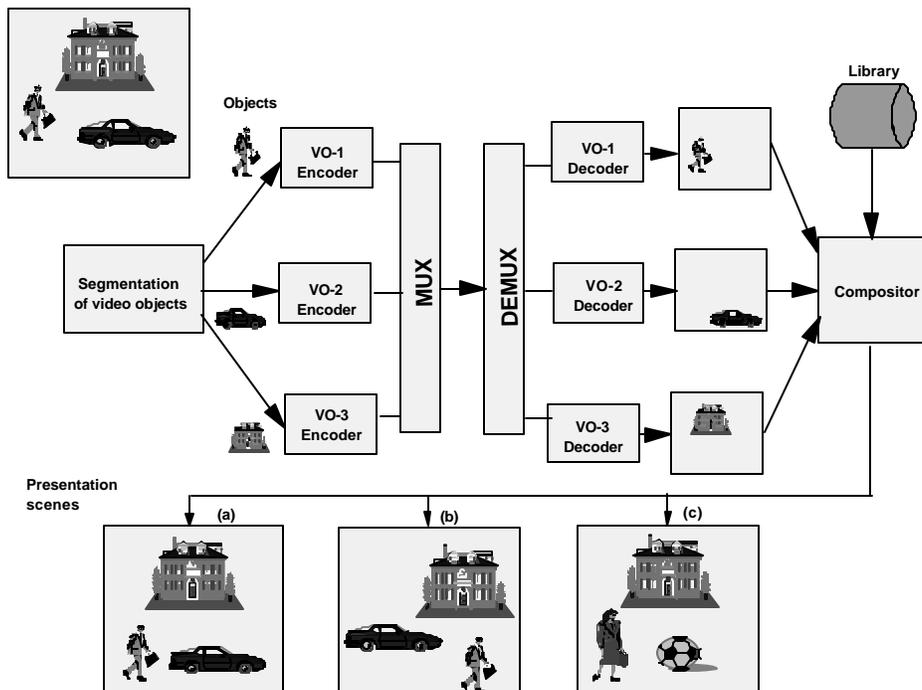


Figure 3. An example of video scene and composition in MPEG-4.

2. VISUAL OBJECTS

Visual objects in MPEG-4 may be composed of natural objects, synthetic objects, or both. Natural objects are pixel based images and video, while synthetic objects are computer-generated images and animations.

2.1 NATURAL VISUAL OBJECTS

The elementary units of a natural visual object are called video objects (VOs). An example of a VO could be a tennis player (without background) or the crowd watching the tennis player. VOs such as a tennis player, where the object is in motion and the background is masked out, are known as Video Object Planes (VOPs). Because the background is omitted, VOPs are essentially irregularly shaped videos as opposed to the full rectangular video images we get with MPEG-1 and MPEG-2. The background image may be further broken down into other VOPs, or a static sprite. A static sprite is a possibly large still image, describing panoramic background [4]. By using it as a sprite, the background only needs to be fully encoded once. Afterwards, the background can be repositioned through eight global motion parameters encoded for each subsequent frame. The decoder will combine VOPs and sprites to recreate the original video. In this way, the amount of data needed to reproduce a scene is reduced (effectively compressing the scene).

Figure 4 shows how VOPs and sprites may be used to construct a single scene [4]. The tennis player in the foreground is segmented from the background as a VOP. The background is then extracted as a sprite prior to encoding. The sprite image is transmitted only once and is stored in a sprite buffer. In each subsequent frame, only the eight camera parameters relevant to the background are transmitted. The VOP is transmitted separately as an arbitrary-shape video object. The receiver can now reconstruct the background for each frame in proper sequence based on the sprite, and can overlay the VOP onto the background to correctly recreate the original scene.

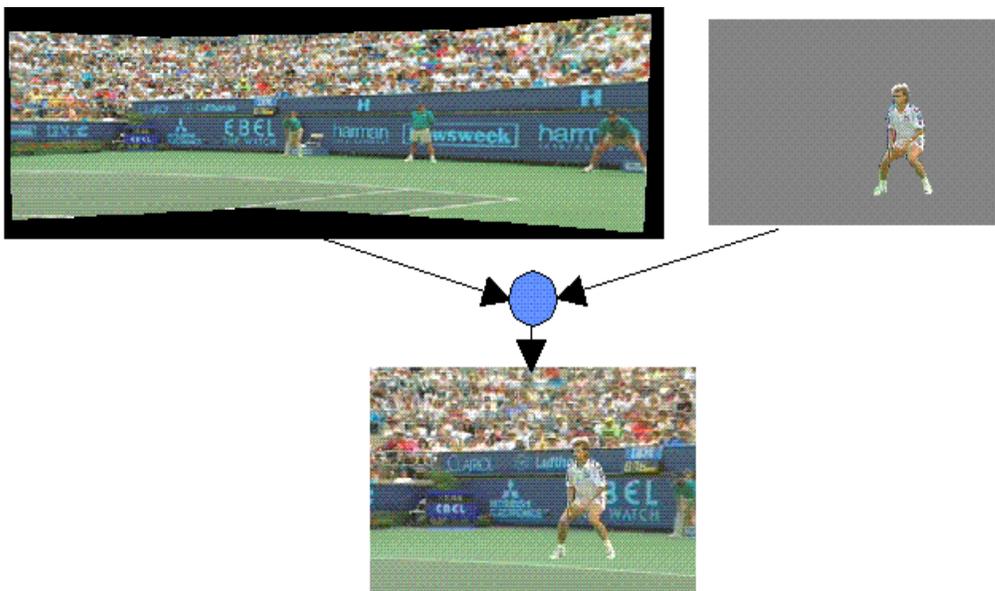


Figure 4. VOP and sprite coding of a video sequence (from [4]).

Multiple VOPs comprising a higher level video object are each coded as a separate layer of the video object, each with its own shape, motion, and texture components. The arbitrary shape comprising an individual layer may be described in two ways. First, for lower quality content or low bit rate environments, binary shape coding may be used. In binary shape coding, a map is used to define whether or not a pixel is part of the object (a pixel is either 'on' or 'off'). The second method of describing a shape is gray scale or alpha shape coding. In this type of coding, each pixel is not merely 'on' or 'off', but is assigned a transparency value. With the added transparency information, VOP layers can be blended more smoothly and will give higher quality content than with binary shape coding.

The ability to blend multiple VOP layers is part of the content-based feature of the MPEG-4 video model. Being content-based, video objects can be separated and separately decoded. This enables scaling of information both spatially (reduction of the size of objects) and temporally (reduction in the amount of information received per unit time). In this way, low bandwidth applications may still be able to obtain meaningful information by scaling back the less important aspects of the information being received.

Coding of each VOP sequence (whether rectangular size or not) is based on the same method already used in the previous MPEG standards [5]. The MPEG-4 coding algorithm encodes the first frame of a VOP as an intra-frame (I-VOP). Subsequent frames are coded using Inter-frame prediction (P-VOPs). Bi-directionally predicted frames (B-VOPs) are also supported and may be used. Each frame is broken into 16x16 pixel macroblocks, which are in turn related to six 8x8 YUV blocks (four luminance – Y1, Y2, Y3, Y4; two chrominance – U, V). This is illustrated by Figure 5.

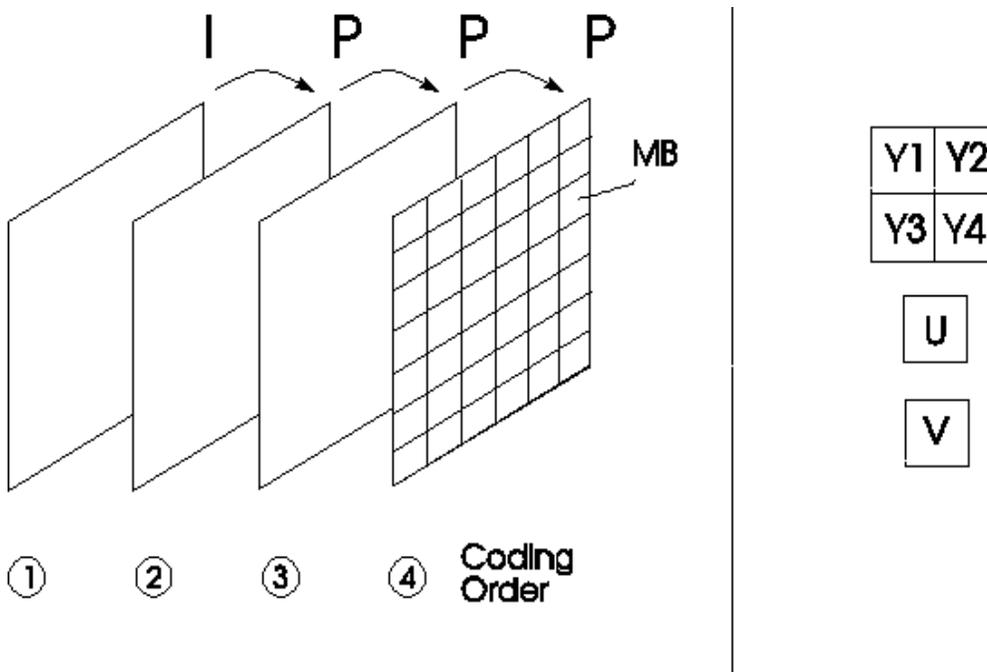


Figure 5. VOP frame coding.

In order to code shape, texture, and motion information for arbitrarily shaped VOPs, a constant size reference window (a multiple of 16x16 pixel blocks) is used as a container for each VOP comprising a scene. The VOP is positioned within the reference window so it is in correct spatial position relative to all the other VOPs comprising the same scene at that same time. Each VOP will use a similar reference window. The VOP shape is mapped onto a shape-adaptive macroblock grid within the reference window. The border of the VOP shape defines the VOP window. A shift parameter is coded to show the position of the VOP window within the reference window. This is illustrated by Figure 6.

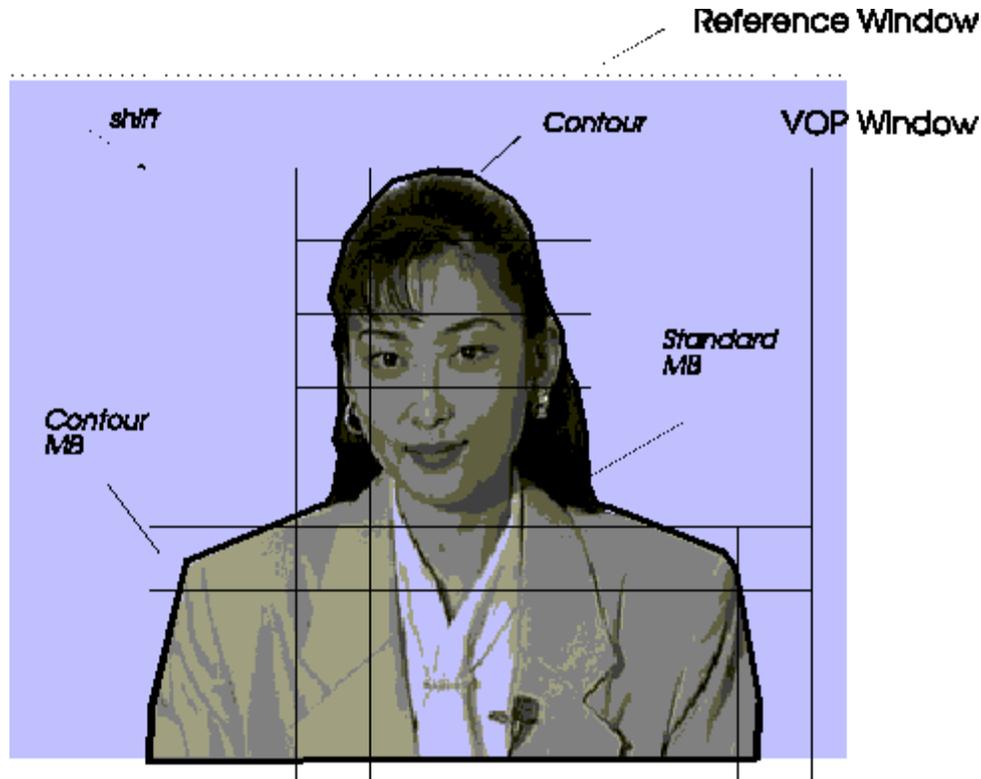


Figure 6. Macroblock grid and reference window (from [5]).

An alternative, more revolutionary way to object-based image coding is to consider the complete image object as the smallest possible entity. The texture, contour, and motion of this object are separately coded [17].

In order to support a wide range of natural video applications, the MPEG-4 standard provides algorithms and tools for very low bit rate applications (typically between 5 and 64 kbit/s), as well as high bit rate applications (typically 64 kbit/s to 10 Mbit/s). Either tool set provides the content-based functionalities (scalability and object manipulation).

2.2 SYNTHETIC VISUAL OBJECTS

Synthetic objects in the MPEG-4 standard may be either two-dimensional or three-dimensional. Two-dimensional objects consist of rectangles, circles, lines, polygons, curves, and 2-D meshes [6]. Three-dimensional objects consist of boxes, cones, cylinders, spheres, and 3-D facial meshes [6]. These objects, both 2-D and 3-D, have the general properties of

transparency, color, and texture. Furthermore, other video objects such as still images or moving video may be mapped onto meshes as texture.

A 2-D mesh is a partition of a 2-D planar region into triangular patches. Animation of the mesh is done by updating motion vectors for the patch vertices, thus giving the image the appearance of motion. Since the motion vectors of the vertices of a triangular patch will most likely be different, the triangle formed by those vertices will be ‘warped’. Likewise, any texture mapped onto that patch will be ‘warped’. Thus, by moving the vertices of the triangles, the patches, as well as their mapped textures, will be ‘warped’. This can be used to create video-like objects which are, in reality, still images mapped to an animated 2-D mesh (e.g., a flag mapped onto a rectangular mesh can be animated to appear as if waving in the breeze). Figure 7 shows an example of a fish mesh [4]. By deforming the mesh, the fish can be made to ‘swim’. Additionally, any texture can be mapped to the mesh (fish, corporate logo, etc.).



Figure 7. 2-D mesh modeling of a fish (from [4]).

Using 2-D mesh animation will provide a large bit rate saving over using motion video since only the motion vector updates are transmitted on a regular basis (the original mesh and texture are only transmitted once).

Currently, the only 3-D mesh supported by MPEG-4 is the 3-D facial mesh. A 3-D mesh is constructed using Face Definition Parameters (FDP). Once constructed, the face is generic with a neutral expression. Now, using Facial Animation Parameters (FAP), the face can be animated. FDPs and FAPs are part of Binary Format for Scenes (BIFS, see section 4.1); the binary scene description language used in MPEG-4. FAPs control animation through a Face Animation Table within the FDPs and through the Face Interpolation Technique. A Face Animation Table provides functional mapping of FAP commands to control points on the mesh. The Face Interpolation Technique determines the effect of multiple FAPs on common control points before applying the animation.

Like 2-D meshes, any texture can be mapped onto the 3-D facial mesh. Usually, however, a facial image is mapped onto the mesh. This is illustrated by Figure 8.

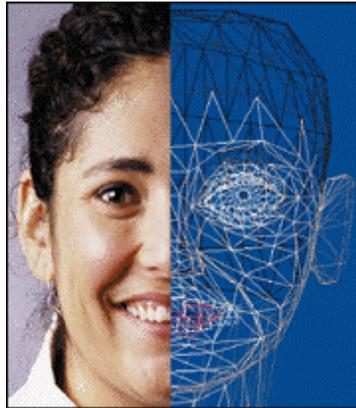


Figure 8. Facial mesh (from [3]).

Synthetic visual objects have view-dependent scalability. This means that the viewing position in a 3-D world is taken into account when determining what data to send. Only that information relevant to objects visible at the viewing position is sent. This may scale down the amount of encoded and decoded data considerably.

2.3 ERROR ROBUSTNESS

To accommodate a wider range of applications, some of which may be more prone to error (such as wireless networks), MPEG-4 provides for robust error recovery through three general mechanisms: resynchronization, data recovery, and error concealment.

Resynchronization allows for the detection of synchronization fields in the bit stream. When an error occurs, only the data from the previous synchronization point to the next synchronization point is considered suspect. The decoder can then discard all of the suspect data or use data recovery techniques to retain part of the data.

Data recovery is used after synchronization has been reestablished so data that might otherwise be lost can be recovered. Generally, the data is encoded in such a way as to make it more resilient to error. One particular method uses the Reversible Variable Length Codes tool to encode the information so that it can be read in both the forward and reverse directions. In this way, data occurring before or after the error can be recovered. Figure 9 illustrates this method of data recovery [4].

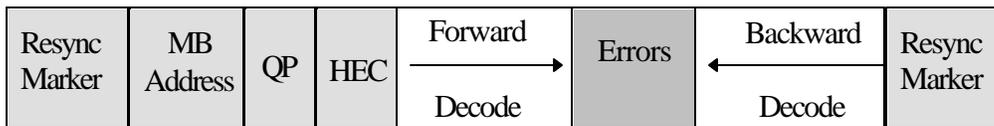


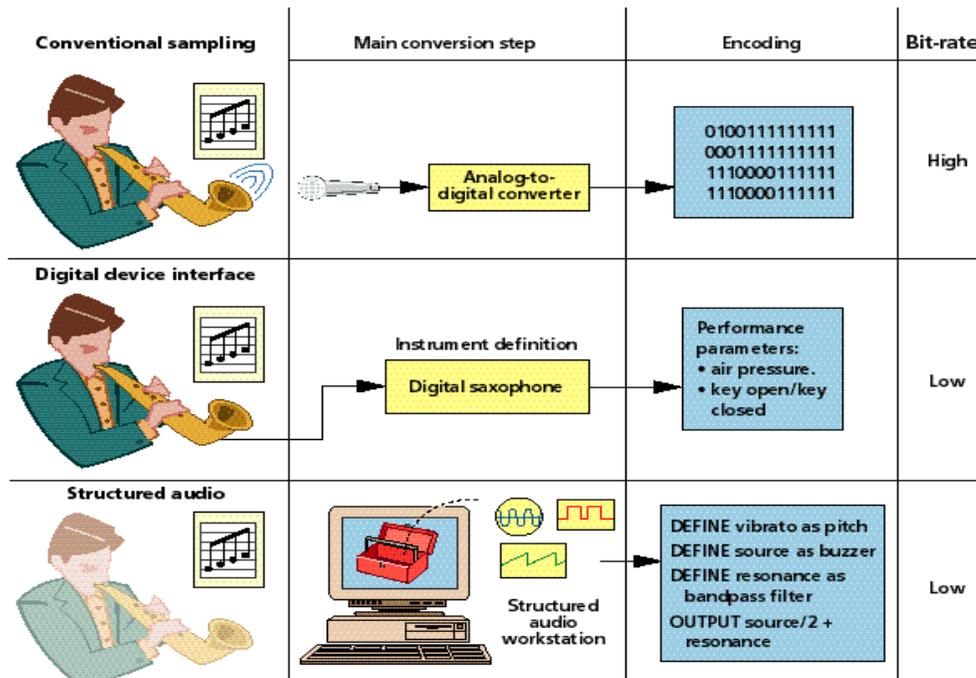
Figure 9. Data recovery using Reversible Variable Length Codes.

Error concealment is used to ‘cover’ for lost data. If lost information can be adequately localized in resynchronization, then lost data may be concealed by using data from previous frames (problem areas will be essentially invisible to the human eye). For enhanced concealment capabilities, texture and motion information could be separated (with an additional resynchronization marker placed between the motion and texture data). Now, if

texture data is lost, it can be recreated using previous texture data along with the current motion data.

3. AUDIO OBJECTS

Audio objects in MPEG-4, like visual objects, may be composed of natural objects, synthetic objects, or both. Natural objects are composed of sounds which naturally occur (are based on analog sound waves). Some natural objects are human speech and orchestral music. Synthetic objects are composed of sounds that are generated (synthesized) based on structured descriptions such as MIDI. MPEG-4 also provides for a more advanced form of synthesized sound with the structured audio format. This format is not itself a method of synthesis, but a way of describing methods of synthesis. This will enable MPEG-4 to accommodate any current or future type of synthesis. Figure 10 illustrates the types of sound handled by the MPEG-4 standard [3]. Natural sound sampling requires a high bit rate, while synthesized sound (whether the simple type used today, or the more complex and higher quality type promised through structured audio) requires a lower bit rate.



Source: Eric Scheirer, Massachusetts Institute of Technology

Figure 10. Types of sound in MPEG-4.

3.1 NATURAL AUDIO

Natural audio encompasses a range of frequencies covering both speech and general audio (music, etc.). In order to achieve high audio quality within this spectrum, the MPEG-4 standard integrates both speech coding techniques and general audio coding techniques into one framework.

MPEG-4 provides tools for coding speech that can work for bit rates from 2 kbit/s to 24 kbit/s (even lower bit rates can be accommodated in variable bit rate mode) [4]. Speech is coded by two algorithms. One, Harmonic Vector eXcitation Coding (HVXC), is a parametric method used for operations at or below 4 kbit/s [4]. The other, Code Excited Linear Predictive (CELP), is used for operating bit rates between 4 kbit/s and 24 kbit/s [4]. CELP uses sampling at 8 kHz and 16 kHz to support narrowband and wideband speech respectively.

General audio is coded at bit rates from 6 kbit/s to beyond 128 kbit/s [3]. This range encompasses everything from a mono signal to high quality stereo. Coding techniques used for general audio include TwinVQ as well as the advanced audio coding (AAC) algorithm from the MPEG-2 standard. AAC provides for efficient compression of audio in the upper bit rate range and provides very high quality audio at effective bit rates which are considerably lower than the MP3 audio format currently used throughout the internet today [3].

Figure 11 shows the general framework of MPEG-4 audio. This framework was defined to allow for scalability (see Section 3.3) while maintaining optimal coverage of the bit rates.

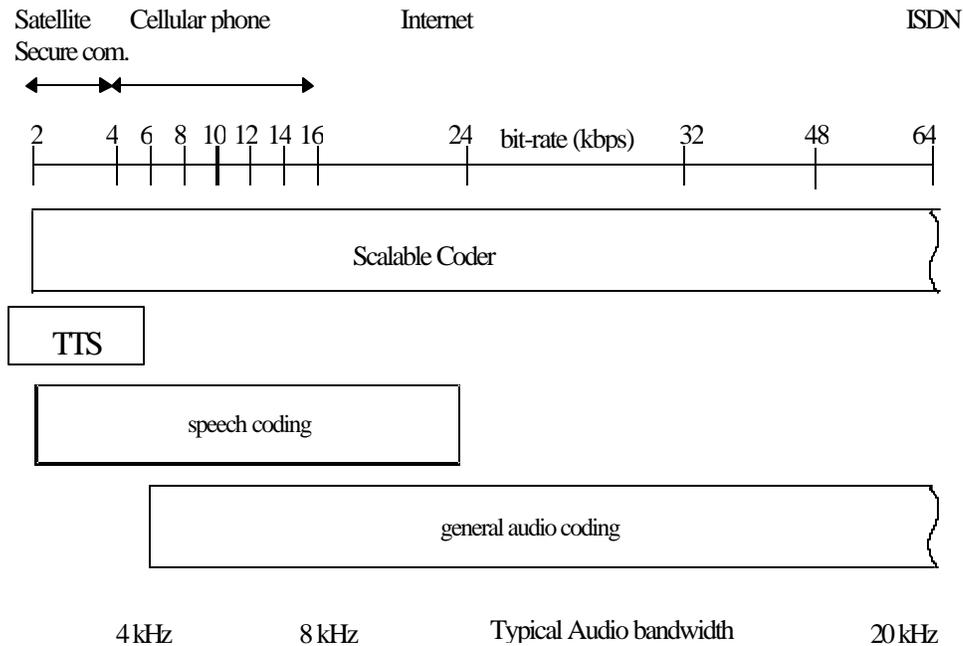


Figure 11. General framework of MPEG-4 audio.

3.2 SYNTHESIZED AUDIO

The MPEG-4 standard provides for several types of input from which to generate synthesized audio. In addition to general sounds, such as music, text input can be converted to speech using the Text-To-Speech (TTS) decoder. Using structured audio tools will allow the delivery of synthesized audio at extremely low bit rates.

TTS coders operate at bit rates between 200 bit/s and 1.2 kbit/s [4]. The generated speech may be synchronized to a facial mesh animation to give the appearance of a talking head. Furthermore, the speech generator can be supplied with prosodic parameters (information

about vocal stress, tone, pitch, accent, cadence, etc.) to facilitate the generation of intelligible, human-like speech, as opposed to flat, droning mechanical speech.

Structured audio uses a synthesis language called SAOL (Structured Audio Orchestra Language) to drive its coding/decoding process. Rather than refer to a specific synthesis process, SAOL describes methods of sound synthesis. Within this method description, instruments can be defined and downloaded. Control over the instruments described in SAOL is handled by a language called SASL (Structured Audio Score Language). With SASL, a musical score can be created using the instrument definition. Since structured audio provides for such detailed sound descriptors for the synthesis (oscillators, filters, etc.), the quality of audio produced through structured audio is very high. For applications that do not require or can not handle the sophisticated synthesis of structured audio, MPEG-4 allows for wavetable bank synthesis (already in use on many PC sound cards).

3.3 ADDITIONAL AUDIO FUNCTIONALITY

The use of objects in MPEG-4 allows audio elements to be considered separately (i.e., each audio object is considered a separate channel). Audio objects can be background music (natural or synthesized), speech, a stereo channel, etc. With multiple audio objects, various channels can be mixed in various ways to produce a different output for each listening situation. Additionally, each object can be manipulated independently of any other object (e.g., an echo can be added to background music without modifying foreground speech).

In order to facilitate a variety of applications pertaining to audio objects, additional audio functionality is provided through the MPEG-4 standard. This functionality pertains to the scalability and manipulation of audio objects as follows [4]:

- *Bit rate scalability* – allows a bitstream to be parsed (either during transmission or in the decoder) into a lower bit rate stream so that meaningful data is not lost.
- *Bandwidth scalability* – a particular case of bit rate scalability where a portion of the frequency spectrum is removed from a bitstream.
- *Encoder complexity scalability* – allows encoders of differing complexity to generate valid, meaningful bitstreams.
- *Decoder complexity scalability* – allows decoders of differing complexity to decode a bitstream. In general, higher quality audio is obtained with more complex encoders and decoders.
- *Speed change* – allows, while in the decoding process, a time scale change without altering pitch.
- *Pitch change* – allows, while in the decoding process, a pitch change without altering the time scale. This can only be done when using parametric or structured audio coding methods.
- *Audio effects* – allows for the processing of decoded audio signals so that functions such as mixing, reverberation, and 3-D spatial positioning of sound can be performed.

4. SCENE COMPOSITION AND MANIPULATION

Once a set of audio-visual objects has been formed, they must be grouped together and synchronized to form a scene. This grouping entails the composition of a scene which, due to the object structure in MPEG-4, can be further manipulated by a user. Creation and manipulation of scenes is accomplished through a special language called BIFS.

4.1 BINARY FORMAT FOR SCENES

The language used by MPEG-4 to describe and dynamically change a scene is specified by a compact binary format known as Binary Format for Scenes (BIFS). With BIFS, objects may be added to a scene, removed from a scene, or modified within a scene. Scene modifications (audio or visual) may be made either in response to a set of pre-defined commands, or in response to user interaction.

BIFS, designed with many of the same concepts used in the Virtual Reality Modeling Language (VRML), has the ability to define 3-D objects (sphere, cube, tube, etc.). Unlike VRML, however, BIFS also has the ability to define 2-D objects (lines, rectangles, etc.), and can be used for real-time streaming. Since BIFS code is binary, its data stream is typically 10 to 15 times shorter than a VRML stream with the same content [3].

BIFS is comprised of four components:

- Semantic elements (scene nodes)
- Binary syntax
- BIFS-Update protocol (control scene changes)
- BIFS-Anim protocol (control animation streams)

BIFS scene nodes contain four types of information:

- Media object attributes (define audio-visual properties)
- Structure of scene graph (layout of media objects in 3-D space)
- Pre-defined animation behavior of the objects
- Animation behavior due to user interaction

The benefits of using BIFS are as follows:

- Integration of 2-D and 3-D media using a single format. Scenes will not need to be designed, encoded, and decoded using multiple formats and associated tools. This makes for greater ease of use, and lower resource utilization.
- Streaming of scene information. Scene components do not have to be completely downloaded before scene presentation can begin.
- Improved compression. Compression is improved since a binary format is used rather than a text format. Typically, a 12 to 1 reduction in needed bandwidth can be obtained by using BIFS to stream animation (as opposed to uncompressed text streaming) [1].

As illustrated by Figure 12, BIFS scenes represent visual and audio objects in 3-D space, with the BIFS scene descriptions falling into sub-categories such as [2]:

- 2-D only primitives
- 3-D only primitives
- 2-D and 3-D scenes layered in a 2-D space
- 2-D and 3-D scenes used a texture maps for 2-D or 3-D primitives
- 2-D scenes drawn in a local two-dimensional plane within a 3-D scene.

4.2 SCENE DESCRIPTION

Audio-visual objects in MPEG-4 require additional information in order to be combined into a cohesive scene. This information, called scene description, is used to determine the placement of the objects both spatially and temporally. Illustrated in Figure 13, the

description follows a hierarchical structure that provides a ‘script’ needed for scene composition. Each node represents either an individual audio-visual object, or a group of such objects. The top node represents the scene, and groups all of the objects together. These nodes expose a set of parameters through which aspects of their behavior (pitch, tempo, frame speed, color, etc.) can be controlled.

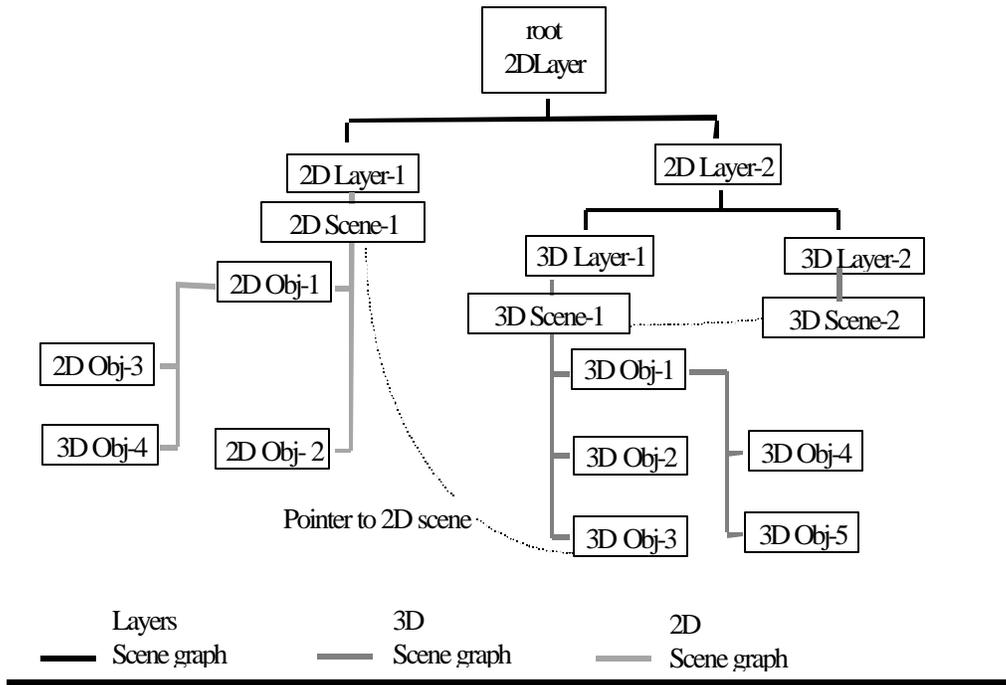


Figure 12. BIFS scene structure (from [2]).

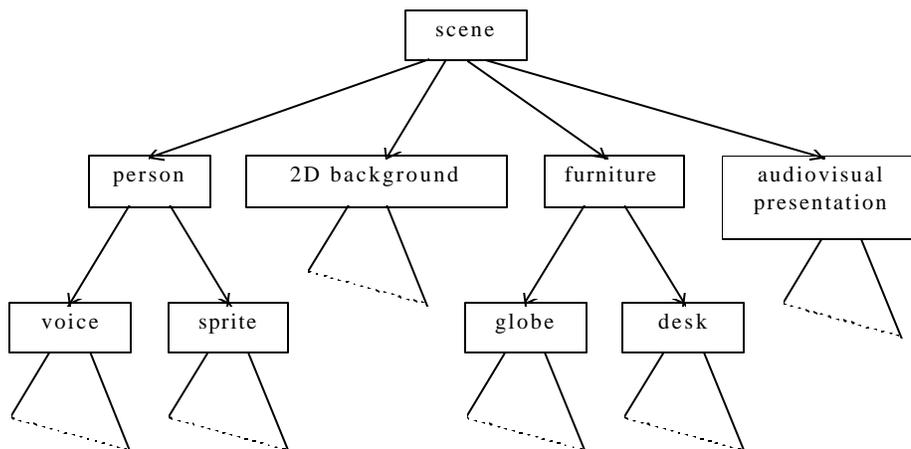


Figure 13. Logical structure of a scene (from [4]).

The information in the scene description is used to place the scene's objects into a common representation space in preparation for rendering to the appropriate media device. This is done through use of local and global coordinate systems. Each media object (node) in the scene description has a local, coordinate system. A local coordinate system is one in which the media object has a fixed spatio-temporal location and scale (orientation and size). Media objects are positioned in a scene by specifying a coordinate transformation from the object's local coordinate system into another coordinate system (global coordinate system) defined by one or more parent nodes in the scene description tree. This is called composition and rendering and is illustrated in Figure 14.

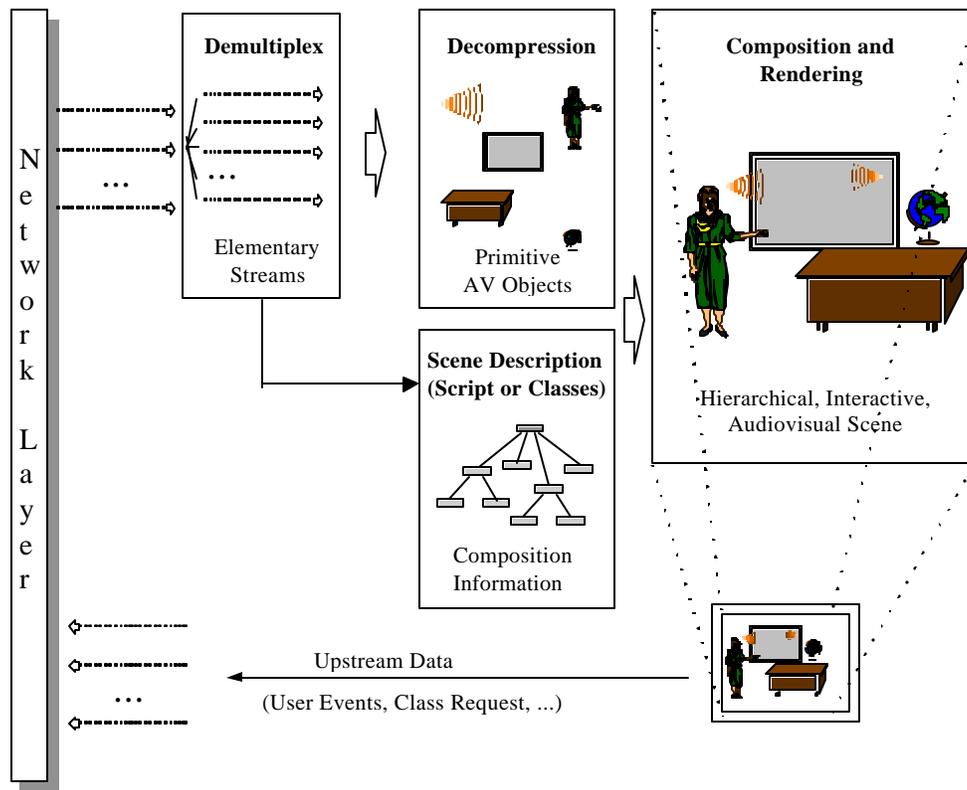


Figure 14. Scene composition (from [4]).

The logical construct used in MPEG-4 to facilitate the transport of media objects is the elementary stream (ES). Elementary streams are produced and consumed by the encoder and decoder respectively, and represent the streaming data between two systems. Some objects, such as background music, may have a single ES. Other objects may have one ES for the basic information and several more for a series of enhancement layers, thus providing scalability. BIFS information used to describe the scene is conveyed in its own ES. This separation of the media object from its scene description provides for reusability of the object (reuse same object with different description) and easy manipulation of the object (modify the description and attributes of the object without having to decode the object's bitstream and change its contents – editing a BIFS bitstream is easy since it is a binary format and not compressed, whereas media object are usually compressed entities).

In order to associate elementary streams to each other and to media objects used in a scene description, object descriptors are used. An object descriptor is a collection of descriptors describing the elementary stream or streams comprising a single media object or scene

description. The descriptors comprising an object descriptor are called elementary stream descriptors. An ES descriptor contains information telling the system how to decode the stream as well as intellectual property identification. Object descriptors, like scene description, are conveyed in a dedicated elementary stream. This allows for the dynamic update and removal of object descriptors or their components as a scene changes. To associate media objects in a scene description to a particular object descriptor, a unique numeric identifier (object descriptor ID) is assigned to each object descriptor. Likewise, an ES descriptor identifies a single stream with a numeric identifier (ES ID), and an optional URL pointing to a remote source for the stream. Access to content is gained through an initial object descriptor made available at session setup. This descriptor points into the scene description and object descriptor elementary streams as illustrated in Figure 15.

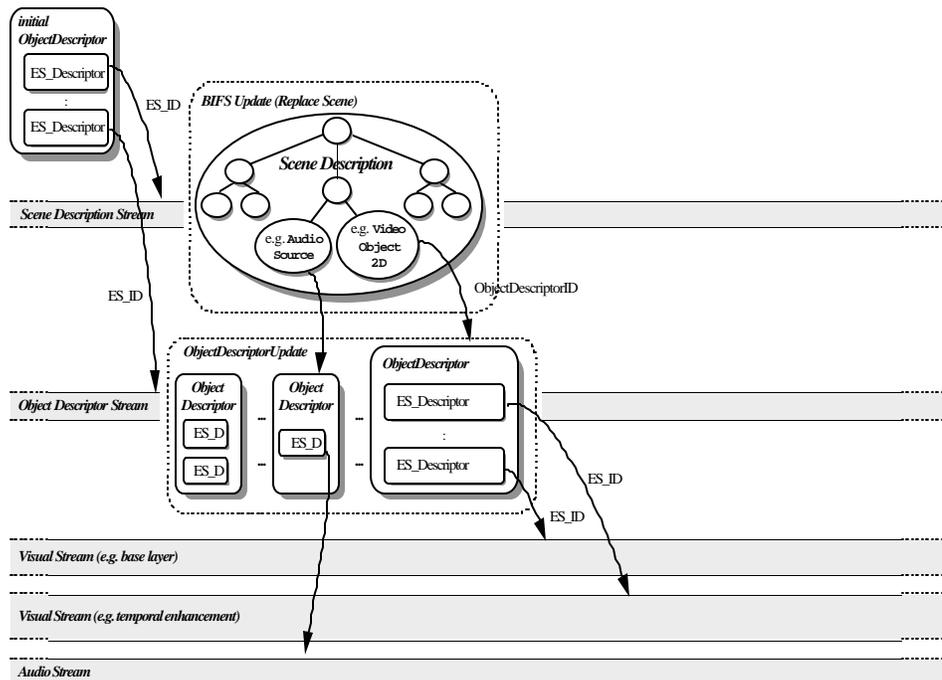


Figure 15. Object descriptor framework (from [2]).

4.3 TIMING

Synchronization of media objects conveyed by one or more elementary streams is dependent upon clock references and time stamps. While time at a receiving terminal is with reference to the terminal's own system time base (STB), each media object has its own reference known as the object time base (OTB). A special time stamp, the object clock reference (OCR), is used to convey the OTB to the decoder. In this way, timing information may be translated from the OTB to the STB for scene synchronization.

To facilitate the synchronization process, data comprising an elementary stream is partitioned into access units. An access unit (AU) is the smallest entity to which timing information can be attributed. Each individually accessible portion of decoder output produced from access units is called a composition unit (CU) – an access unit corresponds to one or more composition units. Composition units reside in a buffered area known as composition

memory. The time at which an access unit must be available for decoding is given by the decoding time stamp (DTS), while the time at which a composition unit must be available for composition is given by the composition time stamp (CTS). At DTS, an AU is instantly decoded with the resulting CUs being placed in composition memory. This is illustrated in Figure 16.

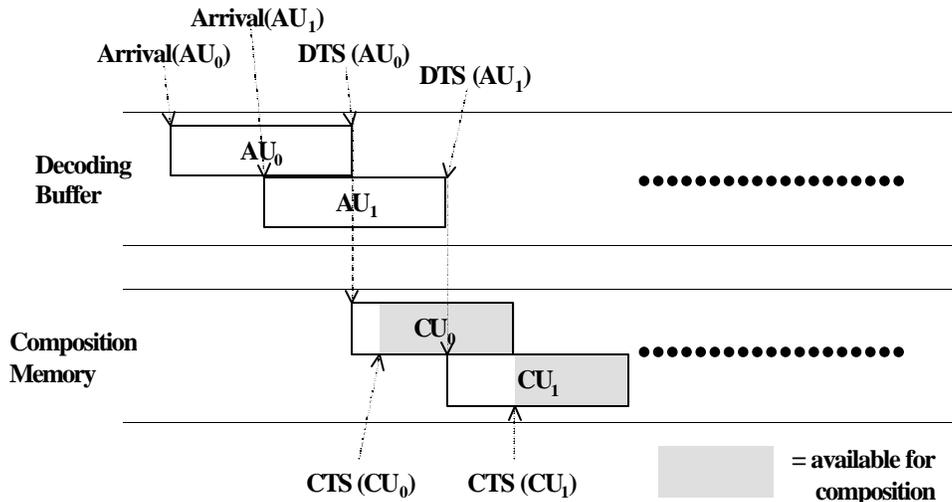


Figure 16. Timing in MPEG-4 (from [2]).

4.4 USER INTERACTION

The content-based and object-oriented nature of the MPEG-4 standard gives way to a high degree of user interaction. Separation of scene description information from media objects allows for manipulation of content attributes at the bitstream without touching the actual object content. Likewise, programmed scene changes can be attached to events associated with objects in a scene (e.g., a mouse click on a spinning globe will cause it to stop spinning).

Some of the possible object manipulations include:

- Change of spatial position
- Change of size
- Change of motion speed
- Addition of optional objects available in composition memory (scaling up)
- Object removal (scaling down)
- Modification of displayed scene area
- Change of pitch
- Change of tempo
- Replacement of scene

Interactivity in MPEG-4 is separated into two major categories: client side and server side. Client side interaction involves content manipulation locally at the user terminal, and is usually performed in response to a user event (mouse click or keyboard stroke). Server side interaction is performed at the transmitting source, and requires additional communication between the sender and receiver. Figure 17 shows an example of scene transformation at the client side [7].

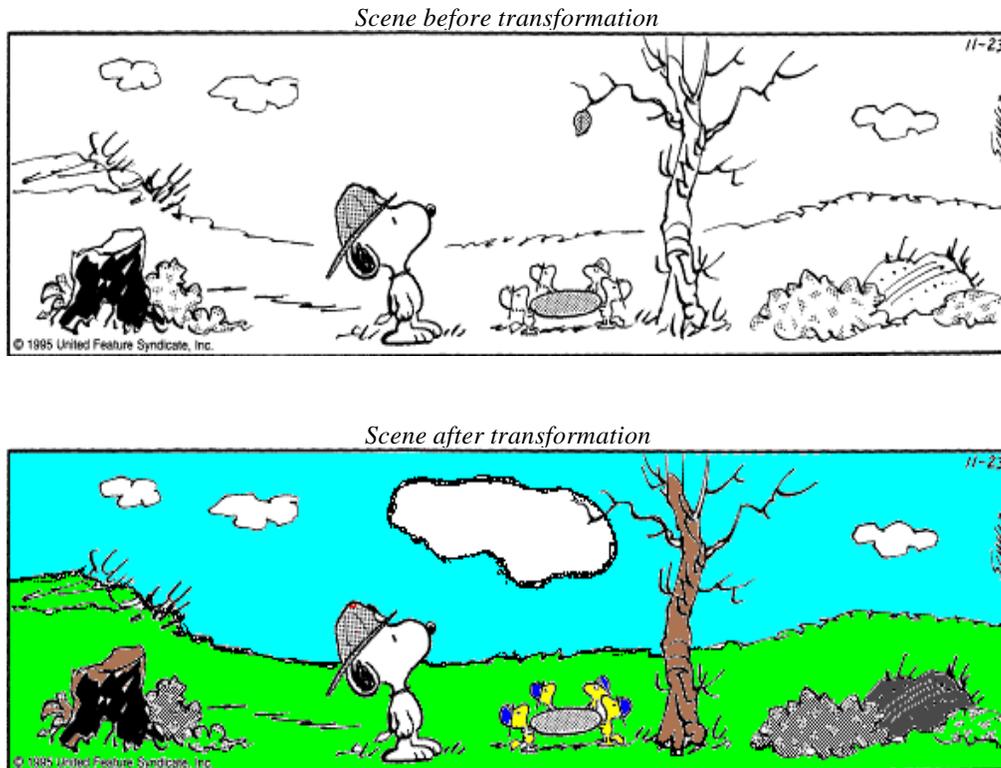


Figure 17. Scene transformation at the client side [7].

5. INTELLECTUAL PROPERTY RIGHTS MANAGEMENT

Intellectual property represents, among others, the ideas, information, and inventions created by an individual or individuals. The legal authority granted designated owners of intellectual property through national and international copyright protection is an intellectual property right. Because of the ability to easily copy electronic information over global networks, a set of tools to ensure intellectual property management and protection (IPMP) is desirable. Although MPEG-4 can not of itself adequately perform IPMP, it does provide for a generic interface to external IPMP tools.

Within MPEG-4, an intellectual property identification is contained within ES descriptors. This identification is dependent upon the content providers. It may be similar to the international standard book number (ISBN) used on books, or it may use a number of key value pairs, such as “Composer/Johan Straus.” The actions performed in carrying out intellectual property protection are in the hands of applications developers. An IPMP framework must be built to take advantage of a standardized MPEG-4 IPMP interface which links MPEG-4 constructs to domain-specific IPMP systems (not standardized in MPEG-4). This interface consists of IPMP descriptors (IPMP-D) and IPMP elementary streams (IPMP-ES). An IPMP-D is an extension of the MPEG-4 object descriptors while an IPMP-ES is an ordinary elementary stream.

IPMP-Ds and IPMP-ESs provide communication between the MPEG-4 terminal and the IPMP system or systems. IPMP-Ds are associated with MPEG-4 objects requiring management and protection. These IPMP-Ds specify which IPMP systems will protect the associated MPEG-4 objects while providing information to these systems about how to protect and manage the objects. Figure 18 illustrates an IPMP framework.

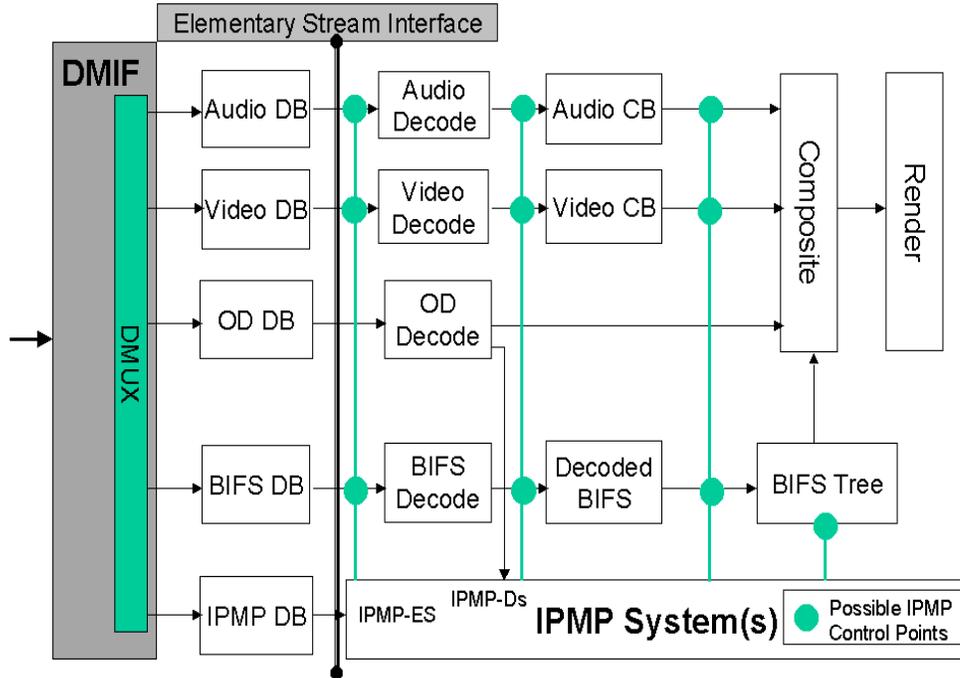


Figure 18. IPMP framework (from [10]).

6. SYSTEMS ARCHITECTURE

The architecture defined by the MPEG-4 standard provides a framework in which the MPEG-4 functionality can be accomplished. That is, it provides the means by which audio-visual information is transported, synchronized, and compressed or decompressed. This is done through a layered approach. First, the TransMux layer provides an interface to the transport medium. Second, the Sync layer handles the synchronization of the streams. Finally, the Compression layer handles the encoding/decoding of the streams. This structure is shown in Figure 19.

To support this architecture, MPEG-4 includes the FlexMux tool and an interface to the DMIF protocol framework. FlexMux (a tool for low delay, low overhead multiplexing of data) can be used to facilitate data transport, while DMIF (Delivery Multimedia Integration Framework) is used to create connections between MPEG-4 sources and destination so that delivery of the necessary information may be accomplished.

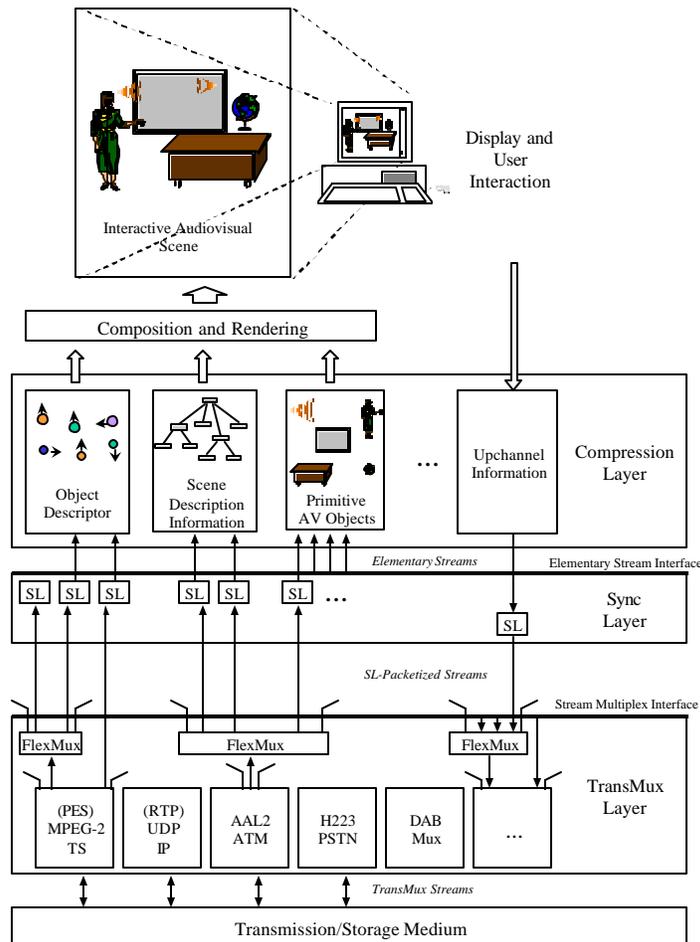


Figure 19. MPEG-4 architecture (from [2]).

6.1 DELIVERY MULTIMEDIA INTEGRATION FRAMEWORK

DMIF is both a protocol and a framework by which multimedia information is delivered. Although DMIF is defined separately (not part of MPEG-4), MPEG-4 does specify an interface to DMIF. DMIF and its interfaces provide the delivery layer. This layer provides transparent information access and delivery irrespective of the technology used. The portion of the TransMux layer dealing with FlexMux channels is considered the DMIF layer. The DMIF layer along with the rest of the TransMux layer encompasses the delivery layer. DMIF works as a session protocol similar to FTP. The essential difference is that FTP returns data while DMIF returns pointers to where information may be obtained.

The DMIF functionality needed for establishing MPEG-4 sessions and access to transport channels is expressed by the DMIF-Application Interface (DAI). This interface lies between the Sync layer and the DMIF layer. When remote communication is necessary, the DMIF Network Interface (DNI) is used. The DNI emphasizes the kind of information DMIF peers need to exchange and requires an additional module to map DNI primitives into corresponding native network signaling messages. Figure 20 illustrates the structure of the delivery layer and its DMIF interfaces.

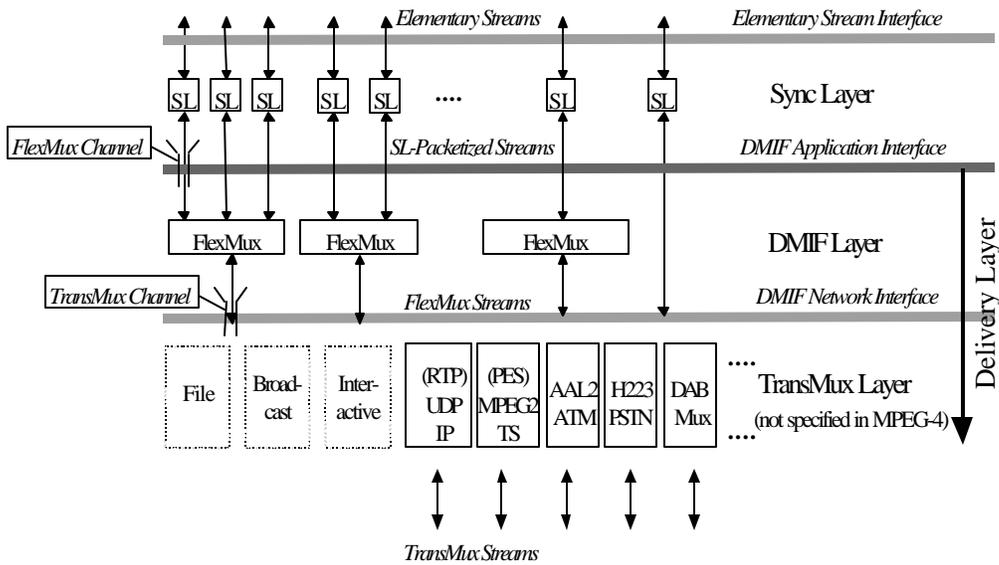


Figure 20. Delivery layer and DMIF interfaces (from [4]).

DMIF takes care of delivery technology details and presents only a simple interface to the MPEG-4 application. The application only interacts with the DAI and is not concerned with the type of transport technology (network, broadcast, and disk or local storage) used. Figure 21 shows how DMIF can be set up to handle the three types of transport technology while leaving all the transport details transparent to the applications.

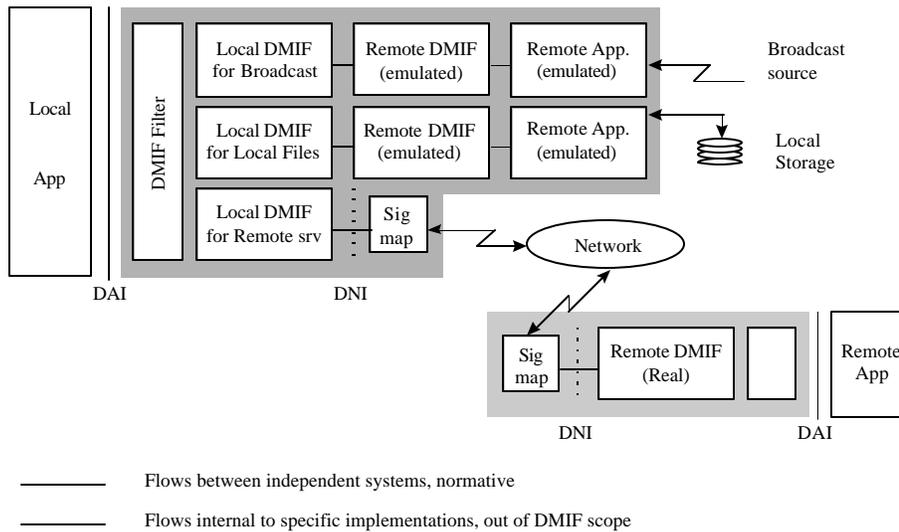


Figure 21. DMIF communication architecture (from [4]).

Establishment of remote connections using DMIF is done by first creating local and remote peer DMIF sessions. Then, using these sessions, a connection between the local and peer application can be made. This is illustrated by Figure 22 as follows [4]:

- The originating application requests the activation of a service to its local DMIF Layer -- a communication path between the application and its local DMIF peer is established in the control plane (connection 1).
- The originating DMIF peer establishes a network session with the target DMIF peer -- a communication path between the originating DMIF peer and the target DMIF peer is established in the control plane (connection 2).
- The target DMIF peer identifies the target application and forwards the service activation request -- a communication path between the target DMIF peer and the target application is established in the control plane (connection 3).
- The peer applications create channels (requests flowing through communication paths 1, 2 and 3). The resulting channels in the user plane (connection 4) will carry the actual data exchanged by the applications.

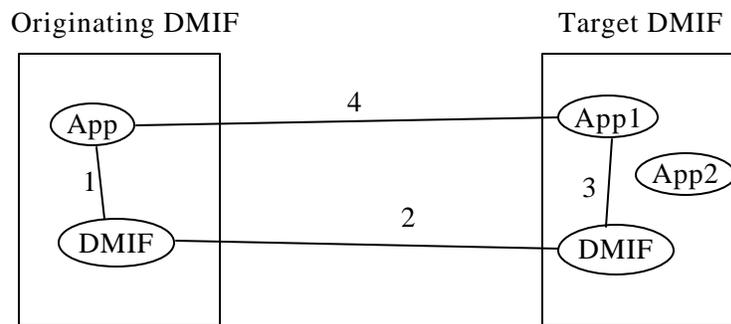


Figure 22. DMIF computational model (from [4]).

6.2 DEMULTIPLEXING, SYNCHRONIZATION, AND DECOMPRESSION

In order to obtain MPEG-4 elementary streams, the information must be extracted from the transmission medium. To do this, the physical data must be demultiplexed as specified by the transport protocol being used. This is outside the scope of MPEG-4 and any multiplexing functionality suitable for MPEG-4 data streams (e.g., ATM, IP, MPEG-2 Transport Stream) may be used. The TransMux layer is a generic term used to abstract this multiplexing functionality. All demultiplexing occurs at the delivery layer (DMIF and TransMux layers). When the transport facility can not sufficiently multiplex MPEG-4 streams, the FlexMux tool is used to provide additional interleaving of data at the transmitter and demultiplexing of the data at the receiver. Data stream interface and use of the FlexMux tool is supported through DMIF.

Once the information is demultiplexed, the delivery layer will pass Sync layer packetized streams to the Sync layer. Sync layer packetized streams are composed of one access unit or a fragment of an access unit. The Sync layer extracts the timing information associated with the access units to enable synchronized decoding and, subsequently, composition of the elementary stream data.

The compression layer reads the elementary streams so that it can form a scene description and decode compressed media objects in preparation for composition and rendering.

7. MPEG-4 VERSION 2

Although the MPEG-4 standard was frozen in October 1998, work has continued for version 2 of the MPEG-4 standard. Version 2 is not meant to replace the current MPEG-4 standard. Rather, it is meant as an enhancement to the standard and is to be fully backward compatible with version 1. This section summarizes the planned enhancements.

The following enhancements to visual aspects are planned for MPEG-4 Version 2:

- Increased flexibility in object-based scalable coding of natural video.
- Improved coding efficiency of natural video.
- Improved error robustness in natural video.
- Coding of multiple views in natural video.
- Addition of 3-D body meshes (extension to facial meshes).
- Coding of generic 3-D meshes.
- Improved 3-D mesh scalability (allow change in the level of detail and allow spatial reduction)

The following enhancements to audio aspects are planned for MPEG-4 Version 2:

- Improved error resilience.
- Improved parameterization of acoustical properties to provide environmental spatialization.
- Reduction in coding/decoding delay.
- Backchannel syntax.
- Small step scalability – allows scalable coding with very fine granularity.
- Parametric audio coding.

The following enhancements to BIFS are planned for MPEG-4 Version 2:

- Multiuser functionality – allows several users to interact with a scene.
- Advanced audio BIFS – enables more natural sound source and environment modeling.
- Added support for 3-D body animation.
- Added VRML compatibility

Other enhancements planned for MPEG-4 Version 2 are:

- Improved delivery capabilities through an expanded DMIF.
- Formalization of an MPEG-4 file format (MP4).
- MPEG-J – Java programming interface

8. INTERNET APPLICATIONS

The ability to scale and separate scenes into component objects makes MPEG-4 a useful tool in the facilitation of applications over the Internet. These applications, which may also apply to other network environments (LANs, WANs, Intranet, etc.) include:

- *Streaming media* – streamed audio and video will be improved through the ability to remove unneeded aspects from the stream (scaling) and the synchronization information imbedded in the stream. In addition, MPEG-4 will allow user interaction

with the stream (e.g., fast forward, fast reverse, pause, and clickable actions to activate video and audio options).

- *Remote content-based storage and retrieval* – because MPEG-4 divides scenes into objects, content-based libraries using MPEG-4 are easier to construct and access than non-MPEG-4 libraries. In other words, identification of desired content is more easily done using the more compact, less cluttered MPEG-4 objects. Thus, Internet applications requiring the retrieval or storage of content-based information are better facilitated through the MPEG-4 standard.
- *Multimedia messaging* – unlike typical text email, multimedia messaging uses audio and video as well. Messages constructed through MPEG-4 will require less bandwidth and may be scaled at the decoder if necessary.
- *Infotainment* – interactive audio-visual presentations (virtual worlds, interactive games, etc.) can be constructed and streamed using MPEG-4. Presentations built using VRML can not be streamed. Also, VRML is a text based language, whereas MPEG-4 uses a binary language (BIFS). Since VRML is a more powerful 3-D presentation tool, complex forms of infotainment will probably use both VRML and MPEG-4 together (the whole VRML construct is itself an object).
- *Interpersonal communications* – MPEG-4 allows for presentation control and scalability of such interpersonal communications as videoconferencing and videotelephony. Using MPEG-4, an additional stream can be added to provide augmented reality. Using augmented reality, a video stream can be added to the audio-video stream used for conferencing so that synthetic video objects can be included in the scene (e.g., synthetic furniture added to an empty office).

9. CONCLUSIONS

In summary, MPEG-4 integrates most of capabilities and features of multimedia into one standard, including live audio and video, synthetic objects, and text, all of which can be combined in real-time and interactively. The content-based, object-oriented nature of the MPEG-4 standard, along with its ability to provide intellectual property rights management and to deploy along a transparent delivery interface should make the standard appealing to a variety of applications. These applications (in addition to those previously listed in Section 8) include:

- Broadcast media
- Interactive storage media – DVD, etc.
- Local content-based storage and retrieval
- Wireless multimedia

Although some reactions to the MPEG-4 standard have been somewhat hostile, in the belief that the standard is meant to supersede MPEG-2 (especially with the HDTV work already done using MPEG-2), MPEG-4 was never meant to replace MPEG-2 [3]. Instead, it has been created to enable new applications (such as good wireless multimedia) and new content types. With its ability to reduce the amount of information needed to create a scene through the use of BIFS, static sprites, VOPs, animations etc., along with its user interactivity and scalability, the MPEG-4 standard should find many uses for a long while.

There is a number of issues in MPEG-4 that are currently under the investigation by researchers in this field. One of the critical issues deals with efficient implementations of MPEG-4 coder. A real-time interactive MPEG-4 encoder has been proposed in [13,16], which uses a cluster of workstations. Authors also described three scheduling schemes for assigning the encoding tasks to the workstations in order to achieve proper load balancing.

Another important topic includes the development of new methods for texture coding for MPEG-4. Some methods for object-based texture coding have been investigated in [17].

An interesting research topic deals with rate control techniques when transmitting MPEG-4 video over the communication networks. Rate control techniques have been intensively investigated for other video coding standards (such as H.261, H.263, MPEG-1, and MPEG-2), and recently for MPEG-4. A new rate control algorithm for multiple video objects in MPEG-4 has been proposed in [18].

References

1. MPEG, "MPEG Systems (1-2-4-7) FAQ, Version 7.0a," Document ISO/IEC JTC1/SC29/WG11 N2527, October 1998.
2. ISO/IEC 14496-1 Systems, "Information Technology – Generic Coding of Audio-Visual Objects," 21 April 1999.
3. R. Koenen, "MPEG-4: Multimedia for Our Time," *IEEE Spectrum*, Vol. 36, No. 1, February 1999.
4. MPEG, "Overview of the MPEG-4 Standard," Document ISO/IEC JTC1/SC29/WG11 N2725, Seoul meeting, March 1999.
5. T. Sikora, Internet Document, "The Structure of the MPEG-4 Video Coding Algorithm," URL: (<http://www.wam.hhi.de/mpeg-video/papers/sikora/fmpeg4vm.htm>).
6. MPEG, "MPEG Requirements," Document ISO/IEC JTC1/SC29/WG11 N2456, Rome Meeting, December 1998.
7. Internet document, "MPEG 4: Coding of Audio-visual Objects," URL: (<http://www.stud.ee.ethz.ch/~rggrandi/mpeg4.html>).
8. MPEG, "DMIF FAQ," Document ISO/IEC JTC1/SC29/WG11 N2313, July 1998.
9. L. Chiariglione, MPEG Internet document, "MPEG-4 FAQs," URL: (http://drogo.csel.stet.it/mpeg/faq/faq_mpeg-4.htm), July 1997.
10. MPEG, "MPEG-4 Intellectual Property Management & Protection (IPMP) Overview & Applications", Document ISO/IEC JTC1/SC29/WG11 N2614, December 1998.
11. L. Chiariglione, MPEG Internet Document, "About MPEG," URL: (http://drogo.csel.stet.it/mpeg/about_mpeg.htm), September 1998.
12. MPEG, "MPEG-4 Applications," Document ISO/IEC JTC1/SC29/WG11 N2724, Seoul Meeting, March 1999.
13. Y. He, I. Ahmad, and M.L. Liou, "Real-Time Interactive MPEG-4 System Encoder Using a Cluster of Workstations," *IEEE Transactions on Multimedia*, Vol. 1, No. 2, June 1999, pp. 217-233.
14. B.H. Haskel, P.G. Howard, Y.A. LeCun, A. Puri, J. Ostermann, M.R. Civanlar, L. Rabiner, L. Bottou, and P. Haffner, "Image and Video Coding – Emerging Standards and Beyond," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 8, No. 7, November 1998, pp. 814-837.
15. T. Sikora, "The MPEG-4 Video Standard Verification Model," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 7, No. 1, February 1997, pp. 19-31.
16. Y. He, I. Ahmad, and M.L. Liou, "A Software-Based MPEG-4 Video Encoder Using Parallel Processing," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 8, No. 7, November 1998, pp. 909-920

17. A. Kaup, "Object-Based Texture Coding of Moving Video in MPEG-4," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 9, No. 1, February 1999, pp. 5-15.
18. A. Vetro, H. Sun, and Y. Wang, "MPEG-4 Rate Control for Multiple Video Objects," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 9, No. 1, February 1999, pp. 186-199.