Submitted to ACM Mobile Networks and Applications Journal, Special Issue on Mobile Multimedia Communications, August 1997 (invited paper).

MPEG-4: An Object-based Multimedia Coding Standard supporting Mobile Applications

Atul Puri

AT&T Laboratories, NSL 3-239
100 Shultz Drive - Middletown
Red Bank, N.J. 07701
apuri@research.att.com

Alexandros Eleftheriadis

Department of Electrical Engineering
Columbia University
New York, N. Y. 10027
eleft@ctr.columbia.edu

Abstract

The ISO MPEG committee, after successful completion of the MPEG-1 and the MPEG-2 standards is currently working on MPEG-4, the third MPEG standard. Originally, MPEG-4 was conceived to be a standard for coding of limited complexity audio-visual scenes at very low bit-rates; however, in July 1994, its scope was expanded to include coding of scenes as a collection of individual audio-visual objects and enabling a range of advanced functionalities not supported by other standards. One of the key functionalities supported by MPEG-4 is robustness in error prone environments. Furthermore, the MPEG-4 standard is being designed to provide solutions for audio coding, video coding, systems multiplex/demultiplex and scene composition in a truly flexible manner.

This paper provides an overview of the current status of the MPEG-4 standard. Section 2 provides a brief overview of the status of related ITU-T standards, since they form a starting basis for video and systems part of the MPEG-4 standard. We then present overview of the MPEG-4 in terms of requirements, tests, video, audio and systems. In sections 4, 5 and 6, with focus on mobile multimedia functionality, we provide discuss the respective coding methods of MPEG-4 Visual, Audio and Systems standards. In section 7 we discuss the issue of profiles, and in section 8 the plans for verification tests are presented. Finally, we summarize the current status of MPEG-4, its planned upgrade to a new version, and the plans for MPEG-7, the next MPEG standard following MPEG-4.

1. INTRODUCTION

The need for mobile communications is ever increasing due to the sense of timeliness and flexibilities it offers. The increasing diversity of mobile applications is now demanding communications not only in the form of speech and data but also with synthetic and natural images and video and is referred to as mobile multimedia. However, multimedia is expensive in the sense of its bandwidth requirement, with video being highly bandwidth intensive. Efficient compression of video is therefore critical to making any multimedia application feasible. The feasibility of mobile multimedia of acceptable quality certainly poses a significant challenge. This is so because wireless channels impose a fairly harsh environment for multimedia communications, and while the goal of compression is to squeeze redundancy out of signals to fit them on limited available bandwidth, the requirements for robust delivery necessitate some amount of redundancy. As in the case for wired or wireless environments, the success of multimedia terminals, products or services [47] depends on many factors, of particular significance is interworking which is facilitated by standardization.

Mobile multimedia applications can be classified into two primary classes, indoor and outdoor. Mobile indoor applications are characterized by lower mobility and higher bandwidth (about 1 Mbit/s or higher) while mobile outdoor applications typically tend to involve higher mobility (including higher speeds) and relatively lower bandwidths (a few kbit/s to a few tens of kbit/s or so). Of course, a number of other applications [18] in between these two extremes also exist. Considering limitations of existing standards when used in mobile applications, a number of researchers have addressed a variety of problems. The focus of this paper is to examine particular considerations for robustness in the state of the art standards being developed. As a passing reference, the ISO MPEG-1 video standard [3] was primarily optimized for coding of noninterlaced video at bitrates of 1.2 to 1.5 Mbit/s and the ISO MPEG-2 video standard [1,4] was primarily optimized for coding of interlaced video at bitrates of 4 to 9 Mbit/s. Furthermore, the MPEG-1 standard assumed a relatively error free channel and the MPEG-2 standard, due to its generic nature, only considered the very basic error resilience techniques such as slice synchronization, intra refresh, and a mechanism to facilitate error concealment, motion vectors for intra coded blocks.

Besides the ISO standards, the ITU-T (formerly, CCITT) has also developed video and audio coding standards. The recently completed ITU-T H.263 standard [4] optimized for video coding at low bitrates of 10 to 24 kbit/s is based on the earlier ITU-T H.261 video standard [2] which was optimized at 64 kbit/s (although it allows a range of 64 kbit/s to 2 Mbit/s). In a general sense, the H.263 standard [5] uses the motion compensated DCT coding framework which is also common to the H.261, the MPEG-1 and the MPEG-2 standards. It however employs a number of features beyond those in H.261 but similar to features in MPEG-1, such as half-pixel motion compensation, as well as several additional modes, such as unrestricted motion vector, advanced 8×8 block prediction, PB-frames, and syntax based arithmetic coding. Incidentally, these modes are options that are negotiated between a decoder and an encoder. However, the ITU-T has continued work on further embellishing H.263 by adding yet many more features and optional modes; the resulting standard, in progress, is referred to as H.263+[22].

The currently ongoing MPEG standard (MPEG-4) [5,6,9,17] was started in 1993 with intended completion by late 1998. Its original focus was modified in July 1994 from that of coding with high efficiency of videophone scenes at very low bitrates, to flexible coding of generic scenes facilitating a number of important functionalities not supported by other standards. Among the functionalities [6] that were considered important for MPEG-4 were content-based coding, universal accessibility (which includes robustness to errors) and good coding efficiency. Further, MPEG-4 video is being optimized for bitrates ranging from about 10 kbit/s to around 1.5 Mbit/s and is expected to be applicable to even higher bitrates. Incidentally, the range of bitrates discussed for MPEG-4 encompasses the bitrates applicable to both indoor and oudoor mobile applications. It is worthwhile pointing out that the MPEG standards [3,4] are essentially decoding standards and thus only specify the bitstream representation and the semantics of the decoding process, in other words, the encoding algorithm is not standardized. The specification of ongoing MPEG-4 standard [25-27] since MPEG-4 is designed to truly be a multimedia standard, goes much beyond that of previous MPEG standards and addresses not only audio coding [27], video coding [26] and multiplexing of coded data [25] but also coding of text/graphics and synthetic images [26] as well as flexible representation of audio-visual scene and composition [25].

Figure 1 shows a high level view of an MPEG-4 terminal [41,43,44]. We use the term “terminal” in a generic sense, including both standalone hardware as well as software running on general purpose computers. A set of individually coded audiovisual object (natural or synthetic) are obtained multiplexed from a storage or transmission medium. They are accompanied with scene description information, that describes how these objects should be combined in space and time in order to form the scene intended by the content creator. The scene description is thus used during composition and rendering, which results in individual frames or audio samples being presented to the user. In addition, the user may have the option to interact with the content, either locally or with the source, using an upstream channel (if available).

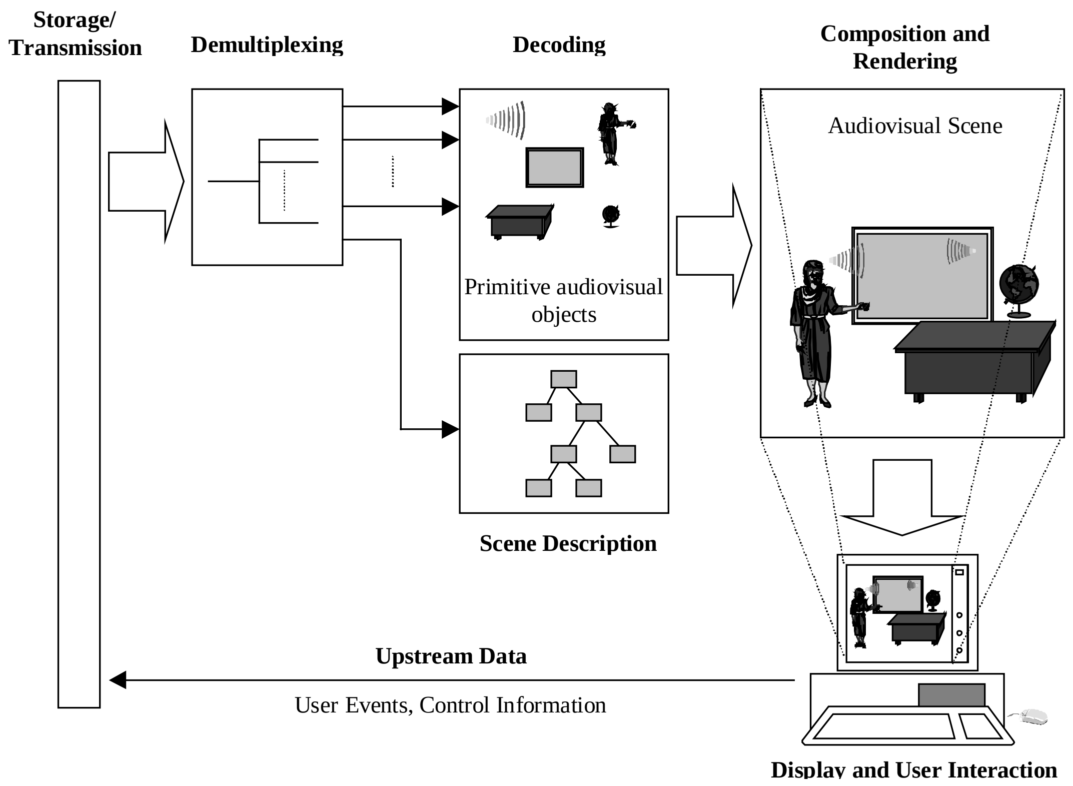

Storage/ Transmission
Demultiplexing
Decoding
Composition and Rendering
Audiovisual Scene
Primitive audiovisual objects
Scene Description
Upstream Data
User Events, Control Information
Display and User Interaction

Figure 1 A high level view of an MPEG-4 terminal

The rest of the paper is organized as follows. In section 2, we present an overview of the ITU-T video and systems standards with emphasis on mobile applications. In section 3, we review the applications, requirements, tests, and the organization of the MPEG-4 standard. Next, section 4 discusses MPEG-4 visual tools with emphasis on error resilience tools. In section 5, MPEG-4 audio standard is briefly discussed. In section 6, MPEG-4 systems is discussed in detail. In section 7 we discuss the issue of profiles, and in section 8 the plans for verification tests are presented. Section 9 discusses the work in progress for version 2 of MPEG-4 video, as well as the plans for the MPEG-7, the next MPEG standard. Section 10 summarizes the key points presented in the paper.

2. RELATED ITU-T STANDARDS

The ITU-T H.263 [5] standard, since it is derived from the ITU-T H.261 standard, is based on the framework of block motion compensated DCT coding. Both the ITU-T H.261 [2] and the H.263 [5] standards, like the ISO MPEG-1 [3] and the MPEG-2 [1,4] standards specify bitstream syntax and decoding semantics. However, unlike the MPEG-1 and the MPEG-2 standards, these standards are video coding standards only and thus do not specify audio coding or systems multiplex, each of which can be chosen from a related family of ITU-T standards to develop applications requiring full systems for audio-visual coding. Also, unlike the MPEG standards, the ITU-T standards are primarily intended for conversational applications (at low bitrates with low delay) and do not include coding techniques that facilitate interactivity with stored data.

The H.324 [19] standard is the multimedia communication standard for low bit-rate circuit switched networks including ordinary analog telephone lines which builds on industry's experience with H.320, the ITU-T standard for ISDN videoconferencing. H.324's architecture consists of multiplexer which multiplexes different media streams into a single stream (H.223) [20], control protocol for capability negotiation (H.245), and audio and video decoding (G.723.1 and H.263). Since, H.263 and H.223 are key components of ITU-T H.324 multimedia communications, we discuss them next.

2.1 H.263 (and H.263+)

The H.263 standard [5] specifies decoding with the assumption of block motion-compensated DCT structure for encoding. This is similar to H.261 which also assumes a block motion-compensated DCT structure for encoding. There are however some significant differences in the H.263 decoding process as compared to the H.261 decoding process, which allow encoding to be performed with higher coding efficiency. For developing the H.263 standard, an encoding specification called the Test Model (TMN) was used for optimizations. TMN's progressed through various iterative refinements, the final test model was referred to as TMN5. The H.263 standard, although it is based on the H.261 standard, it is significantly optimized for coding at low bitrates (of a few tens of kbit/s) while maintaining good subjective picture quality at higher bitrates as well. We now discuss the encoding structure of TMN5 which can provide efficient encoding compliant to H.263 decoding standard.

2.1.1 TMN5 Encoding

Figure 2 shows the block diagram of the simplified encoder allowing video coding as per TMN5. Video coding is performed by partitioning each picture into macroblocks, where a macroblock consists of 16x16 luminance (Y) block (composed of 4, 8x8 blocks) and the corresponding 8x8 chrominance blocks of Cb, and Cr. Each macroblock can be coded is intra (original signal) or as inter (prediction error signal). Spatial redundancy is exploited by DCT coding. Temporal redundancy is exploited by motion compensation which is used to determine the prediction error signal. Block DCT coefficients are quantized by using quant_scale parameter resulting in transmission of quant_index for every nonzero coefficient.

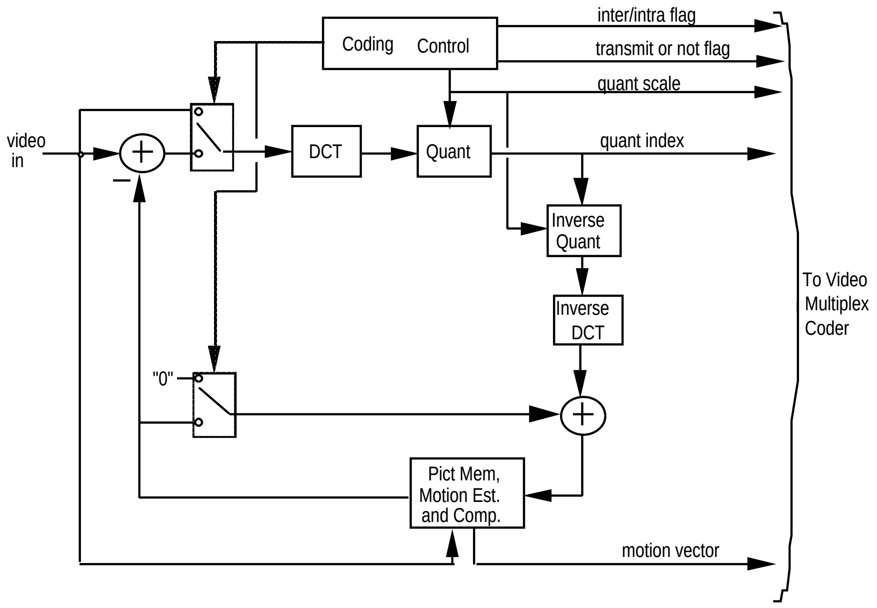


Figure 2 TMN5 Encoder

Besides quant_index for nonzero coefficients, macroblock motion vector and a number of coding control flags and parameters are included in the bitstream generated by the video multiplex coder. In fact the general coding structure of TMN5 described thus far applies not only to H.263 standard but also to all

MPEG video standards. In addition, TMN5 coding also includes motion estimation and compensation with half-pixel accuracy, and bidirectionally coded macroblocks, both of these features are also present in MPEG video standards, which contain additional tools and features as well. Additional features of TMN5 coding are 8x8 overlapped block motion compensation, unrestricted motion vector range at picture boundary, and arithmetic coding; these features are mainly useful for low bitrate applications and were not included in MPEG-1 and MPEG-2 standards.

2.1.2 H.263 Decoding

The H.263 decoder decodes the self-contained bitstream generated by TMN5 or a similar encoder resulting in reconstructed video.

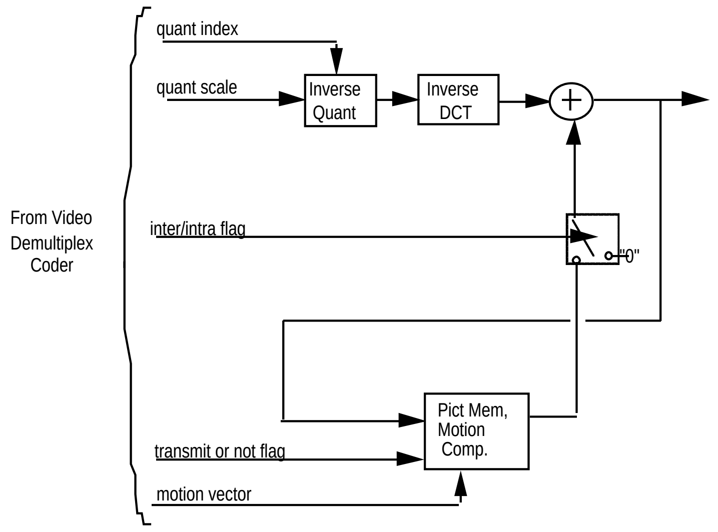


Figure 3 H.263 Decoder

The video decoding algorithm of H.263 is based on H.261 with refinements/modifications to support enhanced coding efficiency. Four negotiable options are supported to allow improved performance. One difference with respect to H.261 is that instead of full-pixel motion compensation and loop filter, H.263 supports half-pixel motion compensation (as discussed during TMN5 encoding), providing improved prediction. Another difference is in Group-of-Block (GOB) structure the header for which is now optional. Furthermore, the four negotiable options of H263 mentioned while discussing TMN5 encoding are as follows.

- Unrestricted Motion Vector mode - This mode allows motion vectors to point outside a picture, with edge pixels used for prediction of nonexisting pixels.
- Syntax-based Arithmetic Coding mode - This mode allows use of arithmetic coding instead of variable length (huffman) coding.
- Advanced Prediction mode - This mode allows use of overlapped block motion compensation (OBMC) with four 8×8 block motion vectors instead of a single 16×16 macroblock motion vector.
- PB-frames mode - In this mode, two pictures, one, a P-picture and the other a B-picture, are coded together as a single PB-picture unit.

2.1.3 H.263+ Features and Modes

As mentioned earlier, the H.263+ [22] standard further adds to H.263 [5], a number of features and negotiable additional modes which are listed as follows.

- Scalability - Spatial, Temporal and SNR scalability;
- Custom Source Formats;

- Advanced Intra Coding (AIC): A mode which improves the compression efficiency for Intra macroblock encoding by using spatial prediction of DCT coefficient values;
- Deblocking Filter (DF): A mode which reduces the amount of block artifacts in the final image by filtering across block boundaries using an adaptive filter;
- Slice Structure (SS): A mode which allows a functional grouping of a number of macroblocks in the picture, enabling improved error resilience, improved transport over packet networks, and reduced delay;
- Reference Picture Selection (RPS): A mode which improves error resilience by allowing a temporally previous reference picture to be selected which is not the most recent encoded picture that can be syntactically referenced;
- Reference Picture Resampling (RPR): A mode which allows a resampling of a temporally previous reference picture prior to its use as a reference for encoding, enabling global motion compensation, predictive dynamic resolution conversion, predictive picture area alteration and registration, and special-effect warping;
- Reduced-Resolution Update (RRU): A mode which allows an encoder to maintain a high frame rate during heavy motion by encoding a low-resolution update to a higher-resolution picture while maintaining high resolution in stationary areas;
- Independent Segment Decoding (ISD): A mode which enhances error resilience by ensuring that corrupted data from some region of the picture cannot cause propagation of error into other regions;
- Alternate Inter VLC (AIV): A mode which reduces the number of bits needed for encoding predictively-coded blocks when there are many large coefficients in the block;
- Modified Quantization (MQ): A mode which improves the bitrate control by changing the method for controlling the quantizer step size on a macroblock basis, reduces the prevalence of chrominance artifacts by reducing the step size for chrominance quantization, increases the range of representable coefficient values for use with small quantizer step sizes, and increases error detection performance and reduces decoding complexity by prohibiting certain unreasonable coefficient representations; and,
- Supplemental Enhancement Information - including chromakey to provide transparency information for implicitly representing shapes of arbitrary regions in pictures.

2.2 H.223 Multiplexer

As mentioned earlier, ITU-T H.324 [19] is a tool-kit standard consisting of component standards, such as V.34 modem, H.223 multiplexer, H.245 control protocol, G.723.1 audio decoder, and H.263 (or H.261) video decoder. We have already discussed H. 263, we now introduce H.223, the multiplexer used to mix audio, video, data and control channels together for transmission on V.34 modem.

ITU-T H.223 [20] multiplexer combines features from time division multiplexers (TDM) and packet multiplexers and new ideas. However, it has less delay then TDM and packet multiplexers and also has less overhead. It is byte-oriented for ease of implementation, can match different data rates using stuffing and also uses a marker for resynchronization recovery. Further, each protocol data unit (PDU) can carry a mix of different data streams in different proportions, hence allowing fully dynamic allocation of bandwidth to the different channels. In addition to the multiplex layer, H.223 also provides a set of three adaptation layers (AL1-AL3). AL1 is primarily intended for variable-rate framed information such as control (e.g. H.245 channel). AL2 is primarily intended for audio (G.723.1), while AL3 is intended for video, such as H.263 or H.261. The ITU-T H.223 Annex A multiplexer has been designed for error prone channels and therefore features a robust packet synchronization and constant packet length. On the protection sublayer, framing, error detection and forward error correction tools using convolutional coding are included.

More specifically, ITU-T H.324 Annex C specifies features of multimedia terminals operating in mobile radio environments, in terms of differences with normal terminals. For instance, three modes are allowed as follows.

- Mode 1: Implement H.223 Annex A [21], the multiplexer for mobile environments, and audio and video codecs of H.324, say, G.723.1 for audio and H.263 (or H.261) for video. The resulting mobile terminal is called H.324M.
- Mode 2: Employ a mobile adaptation layer below H.223. This adaptation layer uses global error correction to H.223 bitstream. The specification of this mode is under study.
- Mode 3: Employ a wireless network based low error protocol below H.223. The specification of this mode is obviously network dependent.

The common procedures to be used when making and using mobile terminals in modes 1, 2, and 3 are quite similar to that in H.324, but for few exceptions: instead of V.34 modem, a wirelesss interface shall be used, V.8 modem shall not be used and the restriction on maximum delay jitter in H.324 is removed. Besides these restrictions, references for mode 1 for H.324M are revised to G.723.1 Annex C instead of G.723, H.223 Annex A instead of H.223 and a subset of codepoints in H.245 are supported. Interworking considerations have also been given. For instance, in mode 1, interoperation with H.324 terminals is expected using an interworking adapter between wireless and GSTN signals; this transcodes between H.223 and H.223 Annex A multiplex. For the case of interworking with modes 2 and 3, no transcoding of multiplexed bitstream is needed since H.223 multiplex is used both for wireless and GSTN signals.

3. MPEG-4 OVERVIEW

MPEG-4 was originally intended for very high compression coding of audio-visual information with at very low bitrates of 64 kbit/s or under. When MPEG-4 video was started, it was anticipated that with continuing advances in advanced (non-block based) coding schemes, for example, in region based and model based coding, a scheme capable of achieving very high compression, mature for standardization would emerge. By mid 1994, two things became clear. First, video coding schemes that were likely to be mature within time frame of MPEG were likely to offer only moderate increase in compression (say, by factor of 1.5 or so) over then existing methods as compared to the original goal of MPEG-4. Second, a new class of multimedia applications were emerging that required increasing levels of functionality than that provided by any other video standard at bitrates in range of 10 kbit/s to 1024 kbit/s. This lead to broadening of the original scope of MPEG-4 to larger range of bitrates and important new functionalities [6]. Basically, three important trends were identified and are as follows.

- The trend towards wireless communications
- The trend towards interactive computer applications
- The trend towards integration of audio-visual data into a number of applications

The focus and scope of MPEG-4 was redefined as the intersection of the traditionally separate industries of telecommunications, computer, and TV/film where audio-visual applications exist. It was felt that the existing standard or emerging audio-visual standards were not adequately addressing the expectations and requirements of these industries in combination leading to incompatible solutions for similar applications. Hence, MPEG-4 was aimed to address these new expectations and requirements by providing audio-visual coding solutions to allow interactivity, universal accessibility and sufficient compression. The mission and the focus statement of MPEG-4 explaining the trends leading up to MPEG-4 and what can be expected in future are documented in the MPEG-4 Proposal Package Description (PPD) document [6]. Figure 4 shows the application areas of interest to MPEG-4 arising at the intersection of the aforementioned industries.

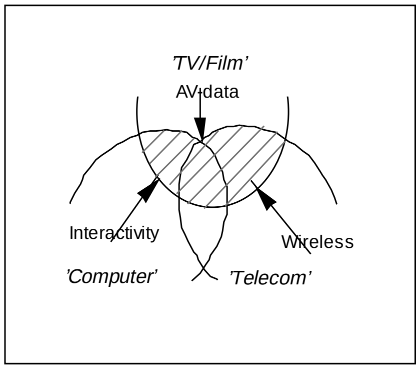


Figure 4 Applications areas addressed by MPEG-4 (shaded region)

To make the discussion a bit more concrete, we now provide a few examples of applications or application classes [43] that MPEG-4 standard is aimed at.

- Internet and Intranet video
- Wireless video
- Video databases
- Interactive home shopping
- Video email, Home movies
- Virtual Reality games, Simulation and training

With revised understanding of goals of MPEG-4 the MPEG-4 work was subsequently reorganized and partitioned into the following subgroups.

- *Requirements* - develops requirements, application scenarios and meaningful clustering of coding tool combinations for interoperability (profiles)
- *Tests* - develops methods for subjective and objective assesment and conducts tests
- *Video* - develops coded representation of moving pictures of natural origin
- *Synthetic and Natural Hybrid Coding (SNHC)* - develops coded representation of synthetic audio, graphics and moving images
- *Audio* - develops coded representation of audio of natural origin
- *Systems* - develops techniques for multiplexing/demultiplexing and presentation of moving images, audio, graphics and data
- *Digital Media Integration Framework (DMIF)* - develops standard and interfaces between digital storage media, networks, servers and clients for delivery bitstreams in networked environments
- *Implementation Studies* - evaluates realizability of coding tools and techniques

Although each of the subgroups has been given its own charter (indicated by their name), they work toward the common goal in a synchronized manner via shared technical documents and joint meetings. Typical process in MPEG development activity starts out with partial collection of requirements, definition of the PPD and a call for technical proposals for evaluation. For instance, in video and audio subgroups the submitted proposals undergo evaluation via subjective testing, objective analysis, and study of implementation aspects; this is often referred to as the *competitive* phase. At the end of the competitive phase the top few proposals are selected and the collaborative phase begins and consists of development of reference coding description which in case of MPEG-4 is called the Verification Model (VM) and is employed for evaluating the performance of competing tools via Core Experiments (CE) and for subsequent optimization of tools. A CE when successful replaces part of the VM (if a similar tool exists) or in other cases simply extends the VM. Thus, VM's undergo an iterative refinement, for instance, VM's in MPEG-4 video have undergone 8 major revisions and the latest one is called VM8. However, the MPEG-4 standard only specifies bitstream and decoding semantics and before becoming an International Standard, undergoes a sequence of iterative Working Drafts, followed by a Committee Draft, a Final Committee Draft and the Draft International Standard. At a recent MPEG-4 meeting, the more mature portions of the ongoing work were labelled as MPEG-4 Version 1, with the remaining portions to be released later in Version 2 or so. In Table 1, we provide the schedule of MPEG-4 Version 1.

Table 1 MPEG-4 Version 1 workplan

Working Draft	**Committee Draft**	**Final Committee Draft**	**Draft International Standard**	**International Standard**
Nov. 1996	Nov. 1997	July 1998	Nov. 1998	Jan. 1999

The MPEG-4 standard (ISO/IEC 14496)[44] is planned to consist of the following basic parts. Other parts may be added when the need is identified.

- ISO/IEC 14496-1: Systems
- ISO/IEC 14496-2: Visual (Natural and Synthetic Video)
- ISO/IEC 14496-3: Audio (Natural and Synthetic Audio)
- ISO/IEC 14496-4: Conformance
- ISO/IEC 14496-5: Software
- ISO/IEC 14496-6: DMIF

3.1 MPEG-4 Functionalities and Requirements

Now that we have some idea of the type of applications MPEG-4 is aimed for we clarify the three basic functionality classes [1,6] that the MPEG-4 standard is addressing. They are as follows:

- *Content Based Interactivity* allows the ability to interact with important objects in a scene. Currently such interaction is typically only possible for synthetic objects, extending such interaction to natural and hybrid synthetic/natural objects is important to enable new audio-visual applications.
- *Universal Accessibility* means the ability to access audio-visual data over a diverse range of storage and transmission media. Due to increasing trend toward mobile communications, it is important that access be available to applications via wireless networks. This acceptable performance is needed over error-prone environments and at low bit-rates.
- *Improved Compression* is needed to allow increase in efficiency in transmission or decrease in amount of storage required. For low bit-rate applications, high compression is very important to enable new applications.

Although we have looked at general classes of functionalities being addressed by MPEG-4 it is desirable to look at specific functionalities that MPEG-4 Version 1 expects to offer; in Table 2 we now show a list of 6 such functionalities [1,6,8] and show their clustering into three functionality classes.

Table 2 Functionalities expected to be supported by MPEG-4 Version 1

Content Based Interactivity
Hybrid Natural and Synthetic Data Coding: The ability to code and manipulate natural and synthetic objects in a scene including decoder controllable methods of compositing of synthetic data with ordinary video and audio, allowing for interactivity.

Improved Temporal Random Access: The ability to efficiently access randomly in a limited time and with fine resolution parts (frames or objects) within an audio-visual sequence. This also includes the requirement for conventional random access.

Content Based Manipulation and Bitstream Editing: The ability to provide manipulation of contents and editing of audio-visual bitstreams without the requirement for transcoding.

Universal Access

Robustness in Error Prone Environments: The capability to allow robust access to applications over a variety of wireless and wired networks and storage media. Sufficient robustness is required, especially, for low bit-rate applications under severe error conditions.

Content Based Scalability: The ability to achieve scalability with fine granularity in spatial, temporal or amplitude resolution, quality or complexity. Content based scaling of audio-visual information requires these scalabilities.

Compression

Improved Coding Efficiency: The ability to provide subjectively better audio-visual quality at bitrates compared to existing or emerging video coding standards.

Besides the new functionalities, MPEG-4 is also supporting the basic functionalities such as synchronization of audio and video, auxillary data streams capability, multipoint capability, low delay mode, coding of variety of audio types, interoperability with other audio-visual systems, support for interactivity, ability to efficiently operate from 9.6 to 1024 kbit/s range, ability to operate in different media environments, and the ability to operate in low complexity mode.

The process of collection of requirements for MPEG-4, although it was started in late 1993, is continuing [41] at present, in parallel with other work items. This is so because development of an MPEG standard is intricate, tedious, thorough and thus time intensive (about 3 to 5 years per standard with overlap between standards). To keep up with marketplace needs for practical timely standards and to follow the evolving trends, the requirements collection process is kept flexible. The major restructuring of MPEG-4 effort in July 1994 to expand its scope was a response to the evolving trends in the marketplace. Evaluating requirements for MPEG-4 is an ongoing complex exercise that uses both, top down (common requirements of related applications) and bottom up approach (related functions provided by a tool, that may be needed in various applications).

The collected requirements are clustered and translated into a set of individual requirements for MPEG-4 Video, Audio, SNHC and System groups as general directions for developing coding methods/tools. In the advanced stages of development, clustering of coding tools takes place to define meaningful profiles (see Sec. 7) that could satisfy application clusters with similar requirements.

3.2 Tests and Evaluation

We now provide details of the testing and evaluation that took place in the competitive phase to determine the potential of the proposed technologies for MPEG-4. We also discuss how the outcome of tests and evaluation was used to initiate the collaborative phase.

3.2.1 Video Tests

The competitive phase was initiated with an open call for proposals in November 1994 (and subsequently revised [8]), inviting technical proposals for the first testing and evaluation [7] which took place in October 1995. A proposal package description (PPD) was developed describing the focus of MPEG-4, the functionalities being addressed, general applications MPEG-4 was aimed at, the expected work plan, the planned phases of testing, information on Verification Models (VM) development, and the time schedule for MPEG-4. The MPEG-4 PPD although started in November 1994 underwent successive refinements until July 1995. In parallel to the PPD development [6], a document describing the MPEG-4 Test/Evaluation Procedures was started in March 1995 and was iteratively refined till July 1995 [7].

The proposers were asked to submit proposals for either complete proposals for formal subjective testing or simply, tools proposals. Since, not all functionalities were tested in the first evaluation, tools proposals were invited for the other functionalities. In a few cases, proposers also used tools submissions as an opportunity to identify and separately submit the most promising components of their coding proposals. Since, tools were not formally tested they were evaluated by a panel of experts and this process was referred to as *evaluation by experts*.

The framework of the first evaluation [7] involved standardizing test material to be used in the first evaluation. Towards that end, video scenes are classified from relatively simple to more complex by categorizing them into three classes: Class A, Class B and Class C. Two other classes of scenes, Class D and Class E were defined; Class D contained stereoscopic video scenes and Class E contained hybrids of natural and synthetic scenes.

Since MPEG-4 is addressing many types of functionalities and different classes of scenes, it was found necessary to devise 3 types of test methods. The first test method was called *Single Stimulus* (SS) and involved rating the quality of coded scene on a 11 point scale from 0-10. The second test method was called *Double Stimulus Impairment Scale* (DSIS) and involved presenting to assessors a reference scene (coded by a known standard) and after a 2 second gap, a scene coded by a candidate algorithm, with impairment of candidate algorithm compared to reference using a 5 level impairment scale. The third test method was called *Double Stimulus Continuous Quality Scale* (DSCQS) and involved presenting two sequences with a gap of 2 seconds in between. One of the two sequences was coded by the reference and the other was coded by the candidate algorithm, and blind tests performed. In DSCQS method, a graphical continuous quality scale was used and was later mapped to a discrete representation on a scale of 0 to 100.

Table 3 summarizes the list of formal subjective tests [7], explanation of each test and the type of method employed for each test.

Table 3 List of MPEG-4 First Evaluation Formal Tests and their explanation

Compression
Class A sequences at 10, 24 and 48 kbit/s: Coding to achieve the highest compression efficiency. Input video resolution is CCIR-601 and although any spatial and temporal resolution can be used for coding, the display format is CIF on a windowed display. The test method employed is SS.
Class B sequences at 24, 48 and 112 kbit/s: Coding to achieve the highest compression efficiency. Input video resolution is CCIR-601 and although any combination of spatial and temporal resolutions can be used for coding, the display format is CIF on a windowed display. The test method employed is SS.
Class C sequences at 320, 512 and 1024 kbit/s: Coding to achieve the highest compression efficiency. Input video resolution is CCIR-601 and although any combination of spatial and temporal resolution can be used for coding, the display format is CCIR-601 on a full display. The test method employed is DSCQS.
Error Robustness
Error Resilience at 24 kbit/s for Class A, 48 kbit/s for Class B, and 512 kbit/s for Class C: Test with high random bit error rate (BER) of 10^{-3}, multiple burst errors with 3 bursts of errors with 50% BER within a burst, and a combination of high random bit errors and multiple burst errors. The display format for Class A and Class B sequences is CIF on a windowed display and for Class C sequences is CCIR-601 on full display. The test method employed for Class A and Class B is SS and that for Class C is DSCQS.
Error Recovery at 24 kbit/s for Class A, 48 kbit/s for Class B and 512 kbit/s for Class C: Test with long burst errors of 50% BER within a burst and a burst length of 1 to 2 seconds. Display format for

Class A and Class B is CIF on a windowed display and Class C is CCIR-601 on full display. The test method employed for Class A and Class B is SS and that for Class C is DSCQS.

Scalability

Object Scalability at 48 kbit/s for Class A, 320 kbit/s for Class E, and 1024 kbit/s for Class B/C sequences: Coding to permit dropping of specified objects resulting in remaining scene at lower then total bit-rate; each object and the remaining scene is evaluated separately by experts. The display format for Class A is CIF on a windowed display and for Class B/C and Class E is CCIR-601 on a full display. The test method employed for Class A is SS, for Class B/C is DSCQS, and for Class E is DSIS.

Spatial Scalability at 48 kbit/s for Class A, and 1024 kbit/s for Class B/C/E sequences: Coding of a scene as two spatial layers with each layer using half of the total bit-rate, however, full flexibility in choice of spatial resolution of objects in each layer is allowed. The display format for Class A is CIF on a windowed display and that for Class B/C/E is CCIR-601 on a full display. The test method employed for Class A is SS, and that for Class B/C/E is DSCQS.

Temporal Scalability at 48 kbit/s for Class A, and 1024 kbit/s for Class B/C/E sequences: Coding of a scene as two temporal layers with each layer using half of the total bit-rate, however, full flexibility in choice of temporal resolution of objects in each layer is allowed. The display format for Class A is CIF on a windowed display and that for Class B/C/E is CCIR-601 on a full display

To facilitate comparison of candidate proposals for MPEG-4 to the existing standards, the later were used as anchors in the subjective tests. In tests involving Class A and B sequences, the H.263 standard (with TMN5 based coding) was used for coding the anchors, likewise for Class C sequences, the MPEG-1 standard was used as anchor. To reduce number of variables that could influence the outcome, the downsampling and upsampling filters were specified for conversion from input formats to lower resolution formats used in coding. To facilitate scalability of arbitrary shaped objects (objects scalability, spatial scalability and temporal scalability), standardized segmentation masks were generated and used by all.

The proposers were required to submit D1 tapes of the coded results, detailed description of their proposal, coded bitstreams and an executable version of decoder software. Although proposers were encouraged to participate in the entire set of tests listed in Table 3, they were allowed to participate in individual tests. About 34 proposers registered for the formal subjective tests. Also, about 40 tools submissions were received for evaluation by experts, but not formally tested. The results of the individual tests and a thorough analysis of the trends were made available [16] during the November 1995 MPEG meeting.

The results [15] of tests in various categories revealed that the anchors performed quite well, usually among the top three or four proposals. In several cases, the statistical difference between top performing proposals was insignificant. In a few specific cases, the new proposals outperformed the anchor or performed similarly but provided additional functionalities. It was also found that since spatial and temporal resolutions were not pre-fixed, there was some difficulty in comparing subjective and objective (SNR) results, due to differences in the choices made by each proposer. It seemed that subjective viewers had preferred higher spatial quality at the expense of temporal resolution. Besides the results from subjective tests, the tools evaluation experts presented their results; about 16 tools or so were judged to be promising for further study.

Soon after the analysis of results of first subjective testing and evaluation in Nov. 1995, the collaborative phase began by collection of tools for purpose of definition of VM and core experiments. A second test and evaluation of proposals, scheduled for mid 1996, was divided into two parts, an extension of first test which was held in Jan. 1996 in the form of evaluation by experts, and , a formal second test which was scheduled for July 1997. A few new proposers participated in Jan. 1996 evaluation and promising new tools were proposed.

3.2.2 Audio Tests

The MPEG-4 Audio also conductive subjective testing, similar to video. Three classes of audio test sequences, Class A, B and C were identified.

- Class A: Single source sequences consisting of a clean recording of a solo instrument.
- Class B: Single source with background sequences consisting of a person speaking with background noise.
- Class C: Complex sequences consisting of an orchestral recording.

All sequences were originally sampled at 48 kHz with 16 bits/sample and were monophonic in nature. For generating reference formats, filters were specified to downsample them to 24, 16 and 8 kHz. A number of bitrates such as were 2, 6, 16, 24, 40 and 64 kbit/s were selected for testing of audio/speech. The first three bitrates are obviously only suitable for speech material. The audio test procedures used were as defined in ITU-R Recommendation 814.

The proposals submitted for testing included variants of MPEG-2 Advanced Audio Coding (AAC), variations of MPEG-1 audio coding and new coding schemes. For specific bitrates, some candidates outperformed the reference coding schemes, although for all combinations tested, no single scheme was the clear winner.

After subjective testing, the collaborative work started and an initial MPEG-4 Audio VM was developed. MPEG-4 Audio development underwent a core experiments process similar to that of MPEG-4 Video development process.

3.2.3 SNHC Tests

The SNHC group started its work much later than the video group. Its focus was primarily on coding for storage and communication of 2D and 3D scenes involving synthetic images, sounds, and animated geometry and its integration into scenes that contain coded natural images/video and sound. Further, it was sought that the coded representation should also facilitate various forms of interactions.

In its Call for Proposals [14] and PPD [15], it sought proposals allowing efficient coding and interactivity in the following areas.

- Compression and simplification of synthetic data representations - synthetic and natural texture, panoramic views, mapping geometry, mapping photometry, animation and deformation
- Parameterized animated models - encoding of parameterized models and encoding of parameter streams
- New primitive operations for compositing of natural and hybrid objects
- Scalability - extraction of subsets of data for time critical use and time critical rendering
- Real-time interactivity with hybrid environments
- Modeling of timing and synchronization
- Synthetic Audio

In the competitive phase, for the purpose of standardized evaluation, a database of test data set was established. The actual evaluation of proposals by a group of experts took place in September 1996. The evaluation criteria was based on the functionality addressed such as, coding efficiency, quality of decoded model, real-time interactivity, anticipated performance in future, and implementation cost. Similar to the tests/evaluations in video, each proposer was required to submit the technical description detailing scope advantages, details and statistics, coded bitstreams and an executable software decoder, a D1 tape showing results, simplification or modification of test data.

At the time, due to participation by a small number of organizations, only a limited number of topics were covered. After the evaluation, the collaborative phase was begun by harmonizing the selected proposals and tools for definition of the first version of SNHC VM and a number of core experiments. The SNHC VM included visual and the audio tools addressing one or more aspects of synthetic data from among, coding, scalability, interactivity and other functionalities. The SNHC effort is thus expected to contribute to tools and algorithms in part 2 and 3 of the MPEG-4 standard. Further, at that time it was expected that MPEG-4 systems would broaden its scope (which it did) to provide the framework needed for compositing decoded natural and synthetic objects in the same scene.

3.3 Video Development

In the period from November 1995 through January 1996, the process of definition of core experiments was initiated. A total of 36 core experiments were defined prior to January 1996 MPEG meeting , another 5 experiments were added at the meeting bringing the total number of experiments [11] to 41. These experiments were classified into a number of topics and four ad hoc groups were formed to coordinate the core experiment process; each ad hoc group was assigned one or more topics as follows.

- Coding Efficiency - prediction, frame texture coding, quantization and rate control.
- Shape and Object Texture Coding - binary and grey scale shape coding, object texture coding
- Robust Coding - error resilience and error concealment
- Multifunctional Coding - bandwidth and complexity scalability, object manipulation, pre and post processing

The result of work of ad hoc group [10] on defining the VM resulted in the first MPEG-4 Video VM (VM1) and was released on 24th January 1996. It supported the following features.

- Coding of arbitrary shaped objects using Video Object Planes (VOPs)
- Coding of binary and grey scale shape of arbitrary shaped objects
- Padding of pixels to fill the region outside of object to full blocks for motion compensation and DCT
- Macroblock based motion-texture (motion compensated DCT) coding derived from H.263
- A mode allowing separation of motion and texture data for increased error resilience

The ad hoc groups undertook the responsibility of producing detailed description of core experiments, finding organizations to independently verify results, and, finalize the experimental conditions. The purpose of the initial series of core experiments was to either offer alternative tools allowing higher coding performance or extra needed functionality compared to VM1. Based on results of core experiments and/or to satisfy additional needed functionality not included in VM1, during the March 1996 MPEG meeting, a number of additional features were added to VM1 and thus VM2 [14] was released on 29th March 1996. The additional features were as follows.

- Bidirectional VOPs derived from combination of H.263 PB-frames mode and MPEG-1/2 B-pictures
- DC coefficients prediction for Intra Macroblocks as per MPEG-1/2
- Extended Motion Vector Range
- Quantization Visibility Matrices as per MPEG-1/2

Since then, there have been six more iterations on the video VM and the process of iterative development and refinement of video VM’s via core experiments has continued. At the July 1997 meeting, a number of mature tools from VM7 [37] have been accepted for MPEG-4 Version 1 which is expected to become the Committee Draft in Nov. 1997. The remaining tools of VM7 with additional tools added at that meeting are will be considered for MPEG-4 Version 2. In section 4, we describe the basic coding methods formed by tools accepted for part 2 of the MPEG-4 Version 1 standard.

3.4 Audio Development

The MPEG-4 Audio coding effort occurred in parallel with the MPEG-2 AAC (formerly, Non Backward Compatible (NBC)) coding effort. The MPEG-2 standard originally had an audio coding mode called backward compatible (BC) mode which as the name suggests was backward compatible with MPEG-1 audio coding. However, at a late stage in MPEG-2 it was discovered that the BC audio coding was rather inefficient compared to non compatible solutions and thus work on NBC mode was begun and overlapped with MPEG-4 schedule. The NBC mode was renamed to be AAC and became a new part of MPEG-2 achieving International Standard status in April 1997 (although it had reached a mature status in mid 1996).

Towards the very low bit-rate end a valid question to ask is why not use the existing ITU-T coders? As an answer to this question, the ITU-T speech coders currently operate at 6.3/5.3 kbit/s (G.723), 8 kbit/s (G.729), 16 kbit/s (G.728), 32 kbit/s (G.721) 48/56/64 kbit/s (G.722). In comparison, MPEG-4 speech coding is being designed to operate at bit rates between 2 - 24 kbit/s for the 8 kHz mode and 14-24 kbit/s for the 16 kHz mode, whereas ITU-T coders do not operate at bitrates as low 2 kbit/s for the 8 kHz mode, or 14-24 kbit/s for the 16 kHz mode. Furthermore, MPEG-4 speech coders are being designed for bitrate scalability, complexity scalability and multi-bitrate operation from 2 - 24 kbit/s. The coding quality of the coder is comparable to that of the ITU coder at corresponding bitrates. MPEG-4 is standardising a speech coder which can operate down to 2 kbit/s. This will be the lowest bit rate international standard. ITU standards do not support this low bit rate. The quality at 2 kbit/s "communication quality" and could be used for usual conversation, and better than FS1016 4.8 kbit/s coder.

Therefore, the MPEG-4 Audio VMs have targeted bitrates from 2 kbit/s to 64 kbit/s; a number of coding schemes are used to cover portions of this range. Besides coding efficiency, content based coding of audio objects and scalability are being investigated. There have been a total of four iterations of audio VM, from VM1 to VM4; the last VM was released in July 1997. In fact, the more mature tools of Audio VM3 have been accepted for the audio part [27] of the MPEG-4 Version 1 standard.

In section 5, we briefly discuss the basic coding techniques accepted for part 3 of the MPEG-4 Version 1 standard.

3.5 SNHC Development

There have been a total of four iterations of SNHC VM, from VM1 to VM4; the last VM was released in July 1997. In fact, the more mature tools of SNHC VM3 have been accepted for the visual part of the MPEG-4 Version 1 standard; the remaining tools have been left in VM4 for consideration for next version of MPEG-4.

In section 4 we describe the SNHC tools expected to be included in the visual part of the MPEG-4 Version 1 standard. In section 5, we discuss SNHC tools expected to be included in the audio part of the MPEG-4 Version 1 standard.

3.6 Systems Development

The Systems layer in MPEG has been traditionally responsible for integrating media components into a single system, providing multiplexing and synchronization services for audio and video streams.

In MPEG-2 [1, 6], for example, these are the primary functionalities, and were designed for two types of transport facilities. The first, Program Stream, is intended for reliable media such as storage devices, and can only carry a single program (combinations of synchronized audio and video streams). It also provided backwards compatibility with MPEG-1 [5]. The second, Transport Stream, is intended for potentially unreliable media and can carry multiple programs. This is shown in Figure 5. The object-based nature of MPEG-4 necessitates a much more complex Systems layer since, in addition to still addressing multiplexing and synchronization, it must also provide for ways to combine simple audio or visual objects into meaningful scenes.

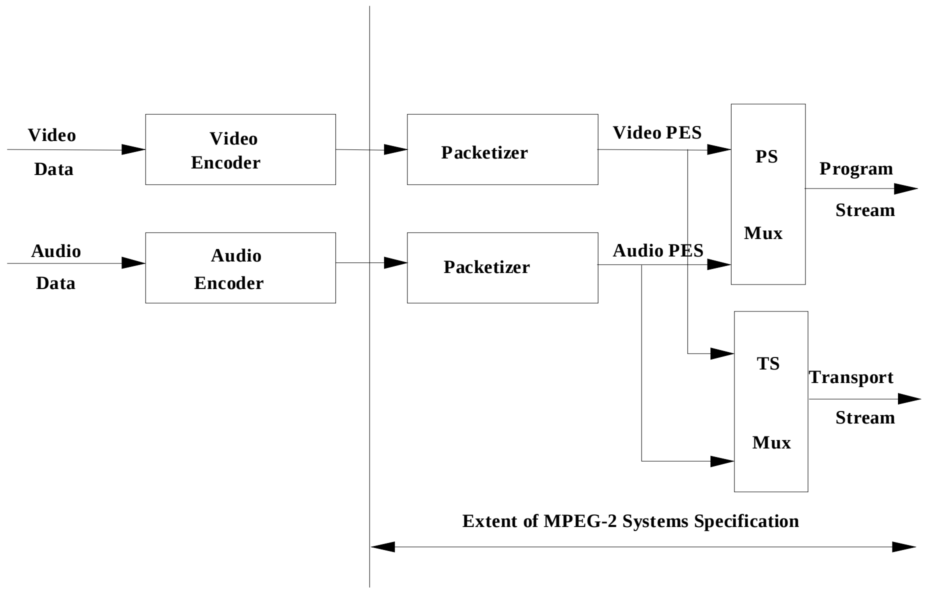


Figure 5 MPEG-2 Systems.

The Systems specification has a long history of evolutionary development [1,24,34-36], starting from the very early stages of MPEG-4 in 1994. Initially, the MPEG-4 project was investigated within the Applications and Operating Environments Group (AOE). The focus of the project was to examine how one could substantially change the paradigm of creation and delivery of audiovisual content, and break away from the limitations of frame- or pixel-based content. New terminal architectures were investigated, favoring programmable architectures that could potentially provide a very high degree of flexibility for application and content developers. It was foreseen that different components of such a terminal could be made to work together by using a special language, termed MPEG-4 Syntactic Description Language (MSDL). This language would describe the syntax of a bitstream, and allow different "tools" to be combined together in various ways to form "algorithms," which would perform particular coding tasks. It is interesting to note that these concepts were articulated before Java became widely known. A Syntactic Description Language, extending C++ and Java, was introduced in November 1995, and underwent several revisions.

Considering the object-based nature of MPEG-4, a key requirement from the System part is the capability to combine individual audiovisual objects in scenes. In late 1995, this was accomplished by using Java [30]. Issues of performance and compliance soon arose. Clearly, it is essential for a content creator to be assured that the content generated will be shown in an identical (or nearly so) regardless of the terminal used, if both such terminals comply to the standard. A three-step approach was adopted, involving three different flexibility levels, as shown in Figure 6. In Level 0, no programmability was allowed. In Level 1, facilities were provided to combine different tools into algorithms, while in Level 2 even individual tools were considered as targets for programmable behavior. The group was also renamed to MPEG-4 System and Description Languages, separating system and syntactic description [24]. After further examination, in late 1996 it was decided that any meaningful operation of Level 1 would require the complexity of implementing a Level 2 system, and hence this intermediate level was eliminated.

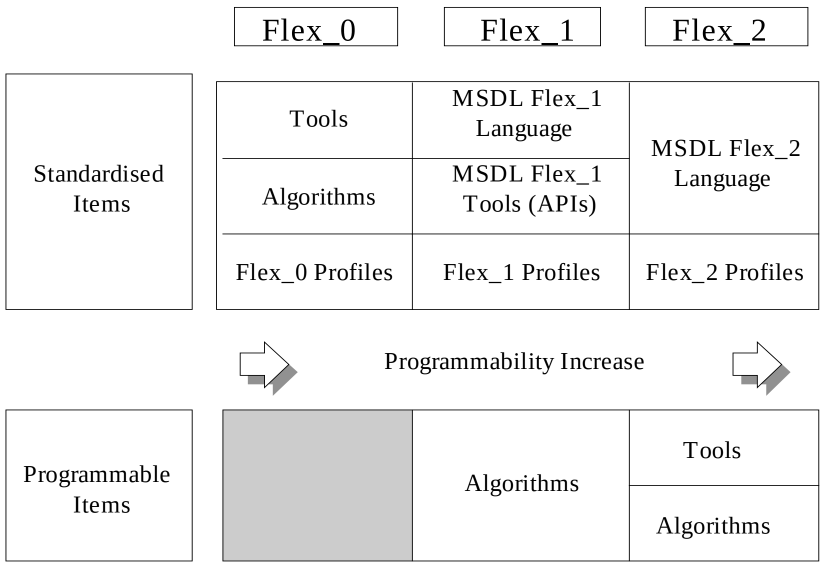


Figure 6 Evolution of MPEG-4 Systems Architecture (1996)

The group subsequently focused on a parametric (bitstream oriented, non-programmable) solution for describing how objects should be combined together. Using the above figure, that would be a Level 0 design. The group also reverted to the use of the traditional term "Systems," reflecting the varied components that it addresses. A programmable approach is still being considered and is discussed in more detail in Section 6.2.4.

In addition to the overall architectural issues, the issue of multiplexing in MPEG-4 also underwent several stages of evolution. The H.223 Annex A multiplexer was used as a basis, including error protection tools (interleaving and ARQ). A key requirement [41], however, for MPEG-4 was the need to be transport-independent. As a result, services that belong to a transport layer were subsequently removed from the set of specified tools, so that efficient implementation of MPEG-4 systems could be performed in a very broad range of environment (broadcast, ATM, IP, and wireless).

3.7 DMIF Development

At a recent MPEG meeting, in recognition of significance of DMF activity has been recognized and DMIF has been given the status of a new group [40]. Previously, DMIF was an ad hoc group operating under the systems group. The charter of DMIF group is to develop standards for interfaces between Digital Storage Media (DSM), networks, servers and clients for the purpose managing DSM resources and controlling the delivery of MPEG bitstreams and associated data. The ongoing work of this group is expected to result in part 6 of the MPEG-4 standard.

4. MPEG-4 VISUAL

The ongoing work on MPEG-4 visual standard specification [26] consists of tools and methods from two major areas - coding of (natural) video and coding of synthetic video (visual part of the SNHC work). We address both these areas; in sections 4.1 through 4.4 we discuss tools and techniques relevant to natural video coding and in sections 4.5 through 4.8 we discuss tools and techniques relevant to synthetic video coding.

4.1 MPEG-4 Video Coding Basics

In this and the next section, we describe the coding methods and tools of MPEG-4 video; the encoding description is borrowed from Video VM7 [37], the decoding description follows [26]. An input video sequence can be defined as a sequence of related snapshots or pictures, separated in time. In MPEG-4, each picture is considered as consisting of temporal instances of objects that undergo a variety of changes such as translations, rotations, scaling, brightness and color variations etc. Moreover, new objects enter a scene and/or existing objects depart, leading to the presence of temporal instances of certain objects only in certain pictures. Sometimes, scene change occurs, and thus the entire scene may either get reorganized or replaced by a new scene. Many of MPEG-4 functionalities require access not only to entire sequence of pictures, but to an entire object, and further, not only to individual pictures, but also to temporal instances of these objects within a picture. A temporal instance of a video object can be thought of as a snapshot of arbitrary shaped object that occurs within a picture, such that like a picture, it is intended to be an access unit, and, unlike a picture, it is expected to have a semantic meaning.

The concept of Video Objects (VOs) and their temporal instances, Video Object Planes (VOPs) is central to MPEG-4 video. A VOP can be fully described by texture variations (a set of luminance and chrominance values) and (explicit or implicit) shape representation. In natural scenes, VOPs are obtained by semi-automatic or automatic segmentation, and the resulting shape information can be represented as a *binary shape* mask. On the other hand, for hybrid (of natural and synthetic) scenes generated by blue screen composition, shape information is represented by an 8-bit component, referred to as *grey scale shape*. In Figure 7, we show the decomposition of a picture into a number of separate VOPs. The scene consists of two objects (head and shoulders view of a human, and a logo) and the background. The objects are segmented by semi-automatic or automatic means and are referred to as VOP1 and VOP2, while the background without these objects is referred to as VOP0. Each picture in the sequence is segmented into VOPs in this manner. Thus, a segmented sequence contains a set of VOP0s, a set of VOP1s and a set of VOP2s, in other words, in our example, a segmented sequence consists of VO0, VO1 and VO2.

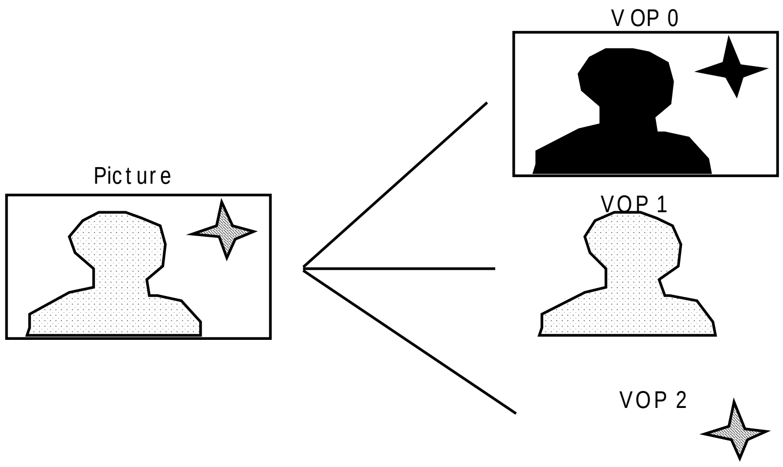


Figure 7 Semantic segmentation of a picture in to VOPs

Each VOs are encoded separately and multiplexed to form a bitstream that users can access and manipulate (cut, paste,..). The encoder sends together with VOs, information about scene composition to indicate where and when VOPs of a VO are to be displayed. This information is however optional and may be ignored at the decoder which may use user specified information about composition.

In Figure 8 we show a high level logical structure of a VO based coder. Its main components are VO Segmenter/ Formatter, VO Encoders, Systems Multiplexer Systems Demultiplexer, VO Decoders and VO Compositor. VO Segmenter segments the input scene into VOs for encoding by VO Encoders. The coded data of various VOs is multiplexed for storage or transmission, following which it is demultiplexed and decoded by VO decoders and offered to compositer, which renders the decoded scene.

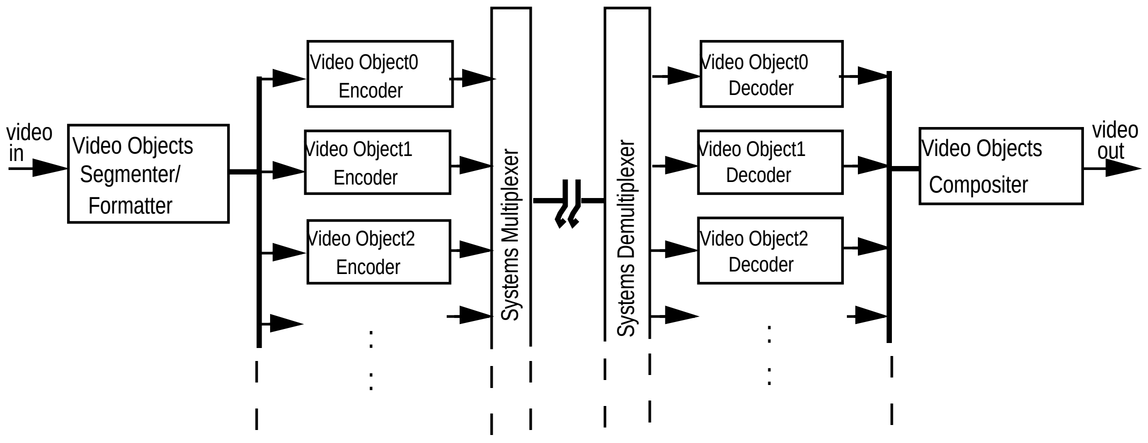


Figure 8 Logical structure of Video Object based codec of MPEG-4 video

To consider how coding takes place in a video object encoder, consider a sequence of VOPs. Now, extending the concept of intra (I-) pictures, predictive (P-) and bidirectionally predictive (B-) Pictures of MPEG-1/2 is to VOPs, I-VOP, P-VOP and B-VOP result. If, two consecutive B-VOPs are used between a pair of reference VOPs (I- or a P-VOPs), the resulting coding structure is as shown in Figure 9.

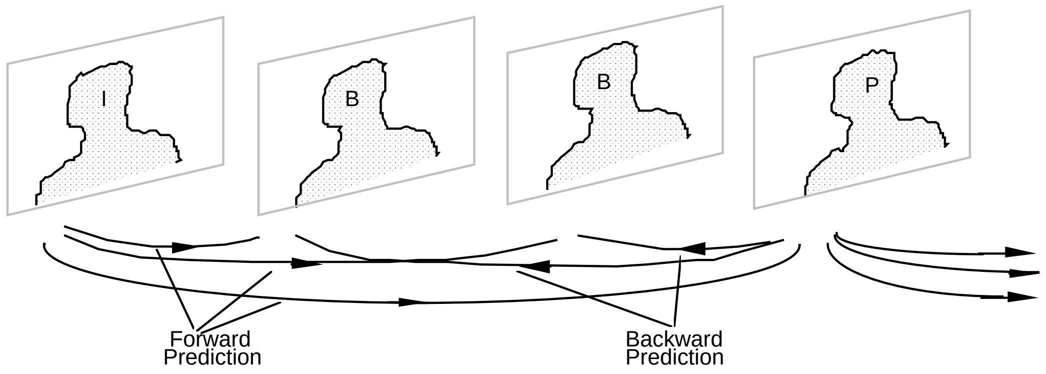


Figure 9 An example prediction structure when using I, P and B-VOPs

In Figure 10 we show the internal structure of the VM based encoder for which codes a number of VOs of a scene. Its main components are: Motion Coder, Texture and Shape Coder. The Motion Coder uses macroblock and block motion estimation and compensation, similar to H.263 and MPEG-1/2 but modified to work with arbitrary shapes. The Texture Coder uses block DCT coding based on H.263 and MPEG-1/2 but much better optimized, further it is also adapted to work with arbitrary shapes. An entirely new component is Shape Coder. The partial data of VOs (such as VOPs) is buffered and sent to Systems Multiplexer.

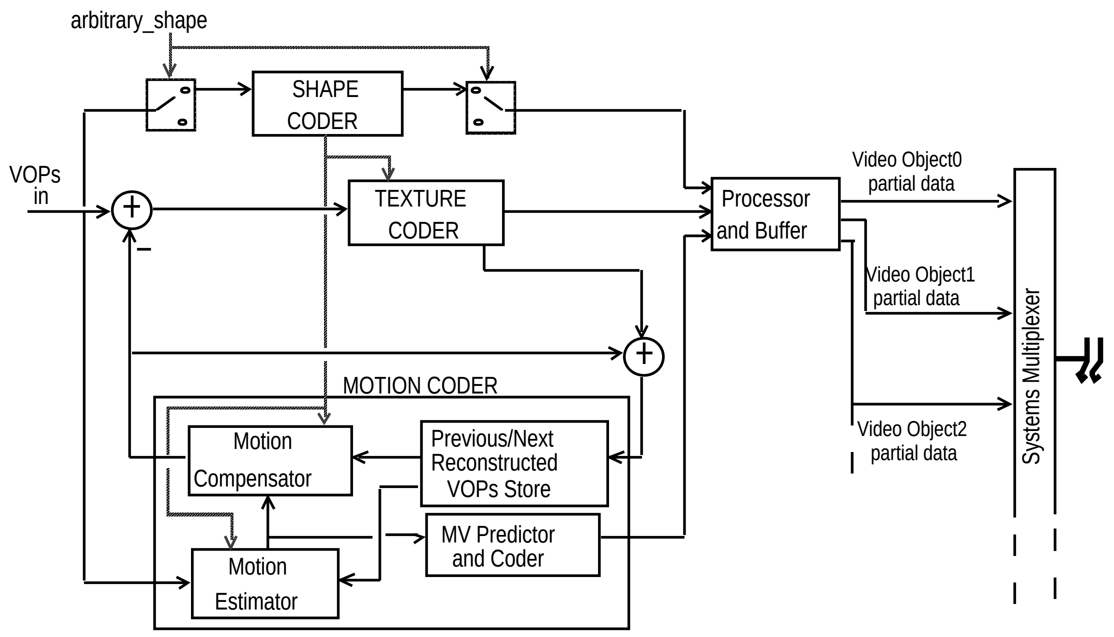


Figure 10 Detailed structure of video objects encoder

From a top-down perspective, the organization of coded MPEG-4 Video data can be described by the following class hierarchy.

- VideoSession: A Video Session represents the highest level in the class hierarchy and simply consists of an ordered collection of Video Objects. This class has only been a place holder for video VM and core experiments work and since composition of objects is now handled by systems, it is not needed.
- VideoObject: A Video Object (2D+time) represents a complete scene or a portion of a scene with a semantic meaning.
- VideoObjectLayer (VOL): A Video Object Layer (2D+time) represents various instantiations of an Video Object. For instance, different VOLs may correspond to different layers, such as in the case of scalability.
- GroupOfVideoObjectPlanes (GOV): Group of Video Object Planes are optional entities and are essentially access units for editing, tune-in or synchronization.
- VideoObjectPlane (VOP): A Video Object Plane represents a snap shot in time of a Video Object. A simple example may be an entire frame or a portion of a frame. different coding methods from MPEG-1/2 such as intra (I-) coding, predictive (P-) coding and bidirectionally predictive (B-) coding can now be applied to VOPs.

The class hierarchy used for representation of coded bitstream described above is shown by the tree structure of Figure 11.

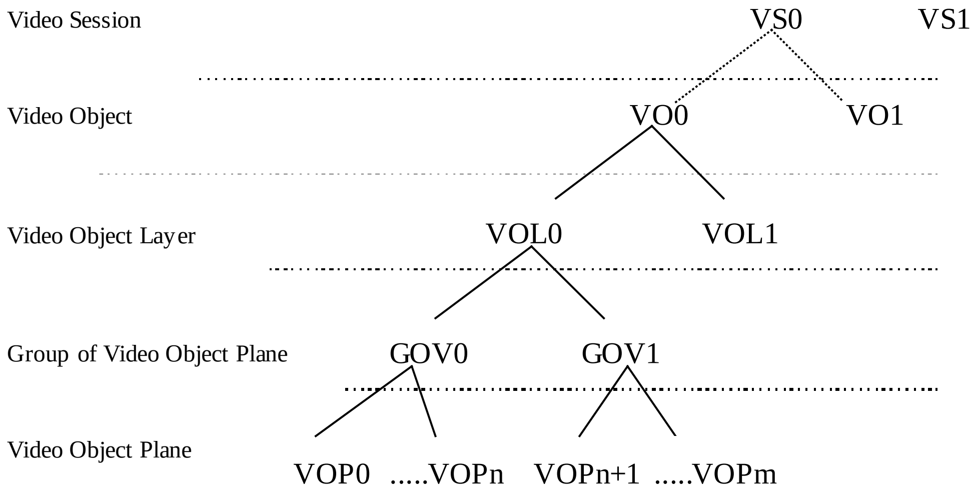


Figure 11 Class hierarchy for structuring coded video data

4.2 Video Coding Details

4.2.1 Binary Shape Coder

Compared to other standards, the ability to represent arbitrary shapes is an important capability of the MPEG-4 video standard. In general, shape representation can be either implicit (based on chroma-key and texture coding) or explicit (boundary coding separate from texture coding). Implicit shape representation, although it offers less encoding flexibility, can result in quite usable shapes while being relatively simple and computationally inexpensive. Explicit shape representation although it can offer flexible encoding and somewhat better quality shapes, it is more complex and computationally expensive. Regardless of the implications, the explicit shape representation was chosen in MPEG-4 video; we now briefly describe the essence of this method [26,37] without its many details.

For each VO given as a sequence of VOPs of arbitrary shapes, the corresponding sequence of binary alpha planes is assumed to be known (generated via segmentation or via chroma-key). For the binary alpha plane, a rectangular bounding box enclosing the shape to coded is formed such that its horizontal and vertical dimensions are multiples of 16 pixels (macroblock size). For efficient coding, it is important to minimize the number of macroblocks contained in the bounding box. The pixels on the boundaries or inside the object are assigned a value of 255 and are considered opaque while the pixels outside the object but inside the bounding box are considered transparent and are assigned a value of 0. If 16×16 block structure is overlaid on the bounding box, three types of binary alpha blocks exist, completely transparent, completely opaque, and partially transparent (or partially opaque). Figure 12 shows an example of an arbitrary shape VOP with a bounding box and the overlaid 16×16 block structure, opaque area is shown shaded whereas transparent area is shown unshaded.

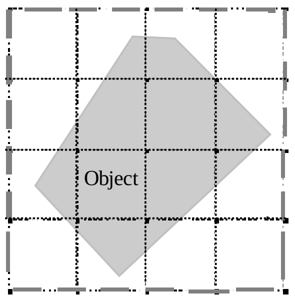


Figure 12 A VOP in a bounding box

Coding of each 16×16 binary alpha block representing shape can be performed either lossy or losslessly. The degree of lossiness of coding the shape of a video object is controlled by a threshold which can take values of 0,16,32,64,..256. The higher the value of this threshold, the increasingly lossy the shape representation; a zero value implies lossless shape coding. Within the global bound of specified lossiness, local control, if needed, can be exerted by, selecting a maximum subsampling factor on a 16x16 binary alpha that results in just acceptable distortion. The estimation of this factor is iterative and consists of using the same subsampling factor in both dimensions and determining the acceptability of resulting shape quality. To be specific, a 4:1 downsampled binary alpha block is used first and if the shape errors are higher than acceptable, a 2:1 downsampled binary alpha block is used next, again if it is found unacceptable, an unsubsampled binary alpha block is used.

Further, each binary alpha block can be coded in intra mode or in inter mode, similar to coding of texture macroblocks. In intra mode, no explicit prediction is performed. In inter mode, shape information is differenced with respect to the prediction obtained using a motion vector, the resulting binary shape prediction error may or may not be coded. The motion vector of a binary alpha block is estimated at the encoder by first finding a suitable initial candidate from among the motion vectors of 3 previously decoded surrounding texture macroblocks as well as the 3 previously decoded surrounding shape binary alpha blocks. Next, the initial candidate is either accepted as the shape motion vector, or is used as the starting basis for a new motion vector search, depending on if the resulting prediction errors of the initial motion vectors are below a threshold. The motion vector is coded differentially and included in the bitstream. Following this procedure, a binary alpha block is assigned a mode from among the following choices.

1. Zero differential motion vector and no inter shape update
2. Nonzero differential motion vector and no inter shape update
3. Transparent
4. Opaque
5. Intra shape
6. Zero differential motion vector and inter shape update
7. Nonzero differential motion vector and inter shape update

Depending on the coding mode and whether it is an I-, P- or B-VOP, a variable length codeword is assigned identifying the coding type of the binary alpha block. The entropy coding of shape data is performed by using the context information determined on a pixel basis to drive an adaptive arithmetic coder. The pixels of binary alpha block are raster scanned, however, a binary alpha block maybe transposed. Next, a context number is determined and is used to index a probability table, and further, the indexed probability is used to drive the arithmetic coder. To determine the context, different templates of surrounding pixels are used for intra and inter coded binary alpha blocks, as shown in Figure 13.

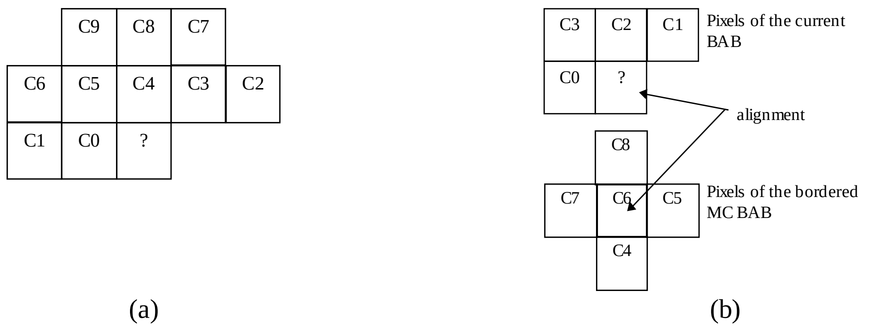


Figure 13 Pixel templates used for (a) intra and (b) inter context determination of a binary alpha block (BAB). Pixel to be coded is marked with ?.

The decoding of binary alpha block follows the inverse sequence of operations with the exception of encoder specific such as motion estimation, subsampling factor determination, mode decision etc, which are readily extracted from the coded bitstream.

4.2.2 Motion Coder

The Motion Coder [26,37] consists of a Motion Estimator, Motion Compensator, Previous/Next VOPs Store and Motion Vector (MV) Predictor and Coder. In case of P-VOPs, Motion Estimator computes motion vectors using the current VOP and temporally previous reconstructed VOP available from the Previous Reconstructed VOPs Store. In case of B-VOPs, Motion Estimator computes motion vectors using the current VOP and temporally previous reconstructed VOP from the Previous Reconstructed VOP Store, as well as, the current VOP and temporally next VOP from the Next Reconstructed VOP Store. The Motion Compensator uses these motion vectors to compute motion compensated prediction signal using the temporally previous reconstructed version of the same VOP (reference VOP). The MV Predictor and Coder generates prediction for the MV to be coded. We now discuss the details of padding needed for motion compensation of arbitrary shaped VOPs, as well as the various modes of motion compensation allowed.

In the reference VOP, based on its shape information, two types of macroblocks require padding, those that lie on the boundary and (depending on encoding choice, some or all of) the other that lie outside of the VOP. Macroblocks that lie on the VOP boundary are padded by first replicating the boundary pixels in the horizontal direction, followed by replicating the boundary pixels in the vertical direction making sure that if a pixel can be assigned a value by both horizontal and vertical padding, it is assigned an average value. Next, the macroblocks that lie outside of the VOP are padded by extending the boundary macroblock pixels, up, down, left and right, and averaging wherever a pixel is assigned a value from more than one directions. The processing for the previous step can be reduced by only padding macroblocks that are outside of the VOP but right next to the boundary pixels.

The basic motion estimation and compensation is performed on 16×16 luminance block of a macroblock. The motion vector is specified to half-pixel accuracy. The motion estimation is performed by full search to integer pixel accuracy vector and using it as the initial estimate, a half pixel search is performed around it. The luminance block motion vector is scaled by a factor of 2 for each component and rounded for use on 8×8 chrominance blocks.

MPEG-4 video, like H.263, supports an unrestricted range for motion estimation and compensation. Basically, motion vectors are allowed to point out of the VOP bounding box, by extending the reference VOP bounding box in all four directions. Further, a larger range of motion vectors is supported for motion vector coding in MPEG-4 as compared to H.263.

Often a single motion vector for a 16×16 luminance block does not reduce the prediction errors sufficiently or when dealing with boundary macroblocks, motion vectors can be sent for individual 8×8 blocks. Further, the 8×8 block motion vectors are used to generate overlapped block motion compensated prediction. Both, the 8×8 block motion compensation and overlapped motion compensated prediction are referred to as advanced prediction in H.263 and are adapted in MPEG-4 to work with arbitrary shaped VOPs.

An intra versus inter coding decision is performed to determine if motion vector(s) need to be sent for the macroblock being coded, further, a decision is also performed to determine if 16×16 or 8×8 block motion vectors will be sent for the macroblock being coded. All motion vectors are coded differentially using median of neighboring three decoded macroblock (or block in case of 8×8 coding) motion vectors as the prediction.

As mentioned earlier, a B-VOP is a VOP which is coded bidirectionally. For example, macroblocks in a B-VOP can be predicted using the forward, the backward or both using the forward and backward motion vectors; this has similarities to MPEG-1/2 in which B-pictures can use such motion vectors. However, MPEG-4 video also supports an H.263 based mode for motion compensation, this is referred to as the direct mode. In direct mode, the motion vector for a macroblock in a B-VOP is obtained by scaling of the P-VOP motion vector, and further correcting it by a small (delta) motion vector. The actual motion compensation mode to be used for a macroblock is decided taking into account the motion compensated prediction errors produced by various choices and the coding overhead of any additional motion vectors. All motion vectors (except delta) are coded differentially with respect to motion vectors of the same type.

MPEG-4 also supports efficient coding of interlaced video. It combines the macroblock based frame/field motion compensation of MPEG-2 with the normal motion compensation of MPEG-4, resulting in overall improved motion compensation. Furthermore, it allows motion compensation of arbitrary shaped VOPs of interlaced video whereas MPEG-2 only supports rectangular pictures of interlaced video.

4.2.3 Video Texture Coder

The Texture Coder [26,37] codes the luminance and chrominance variations of blocks forming macroblocks within a VOP. Two types of macroblocks exist, those that lie inside the VOP and those that lie on the boundary of the VOP. The blocks that lie inside the VOP are coded using DCT coding similar to that used in H.263 but optimized in MPEG-4. The blocks that lie on the VOP boundary are first padded and then coded similar to the block that lie inside the VOP. The remaining blocks are transparent (they lie inside the bounding box but outside of the coded VOP shape) and are not coded at all.

The texture coder uses block DCT coding and codes blocks of size 8×8 similar to H.263 and MPEG-1/2, with the difference that since VOP shapes can be arbitrary, the blocks on the VOP boundary require padding prior to texture coding. The general operations in the texture encoder are: DCT on original or prediction error blocks of size 8×8, quantization of 8×8 block DCT coefficients, scanning of quantized coefficients and variable length coding of quantized coefficients. For inter (prediction error block) coding, the texture coding details are similar to that of H.263 and MPEG-1/2. However, for intra coding of texture data, a number of improvements are included. We now discuss the quantization for intra and inter macroblocks, followed by coefficient prediction, scanning and entropy of intra macroblocks, and finally the entropy coding of inter blocks.

Typically, the DC coefficients of DCT of blocks belonging to an intra macroblock, are scaled by a constant scaling factor of 8, however, in MPEG-4 video, a nonlinear scaler [38] as per Table 4 is used to provide a higher coding efficiency while keeping the blockiness artifacts under the visibility threshold. The characteristics of nonlinear scaling are different between the luminance and chrominance blocks and further depends on the quantizer used for the block.

Table 4 Nonlinear scaler for DC coefficients of DCT blocks

Component	dc_scaler for Quantizer (Qp) range			
	1 through 4	5 through 8	9 through 24	25 through 31
Luminance	8	2Qp	Qp+8	2Qp-16
Chrominance	8	(Qp+13)/2		Qp-6

MPEG-4 video supports two techniques of quantization, one referred to as the H.263 quantization method (with deadzone for intra and inter), and the other, the MPEG quantization method (no deadzone for intra but uses deadzone for inter, and intra and inter quantization matrices). Further, the quantization matrices are downloadable like in MPEG-1/2, but with the difference that it is possible to update matrices partially.

Unlike H.263, the quantized intra DC coefficients are predicted [38] with respect to 3 previous decoded DC coefficients, for example, quantized DC coefficients of blocks A, B and C when predicting quantized DC value for block X in Figure 14. Although MPEG-1/2 also allows prediction of DC coefficients, however the gradient based prediction of MPEG-4 is more effective. In computing the prediction for block ‘X’, if the absolute value of a horizontal gradient ($|QDC_A - QDC_B|$) is less than the absolute value of a vertical gradient ($|QDC_B - QDC_C|$), then the QDC value of block ‘C’ is the prediction, else QDC value of block ‘A’ is used as prediction. This process is independently repeated for every block of an intra macroblock using horizontally and vertically adjacent blocks. Further, the procedure is identical for luminance and chrominance blocks.

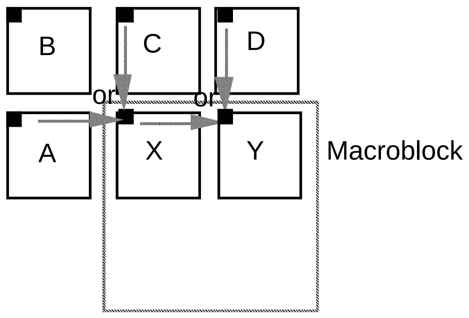


Figure 14 Prediction of DC coefficients of blocks in an intra macroblock

Not only the DC coefficients of intra blocks are predicted and coded differentially, so are some of the AC coefficients [38]. In particular, on a block basis, either the first row or the first column of AC coefficients of DCT blocks of each intra macroblock are predicted. The direction (horizontal or vertical) used for prediction of DC coefficient of a block is also used for predicting the corresponding first column or row of AC coefficients. The prediction direction can differ from block to block within each intra macroblock. Further, the AC coefficient prediction can be disabled for a macroblock when it does not work well. Figure 15 shows the prediction of quantized AC coefficients belonging to the first column or the first row of block X from the corrresponding quantized AC coefficients of block A or block C.

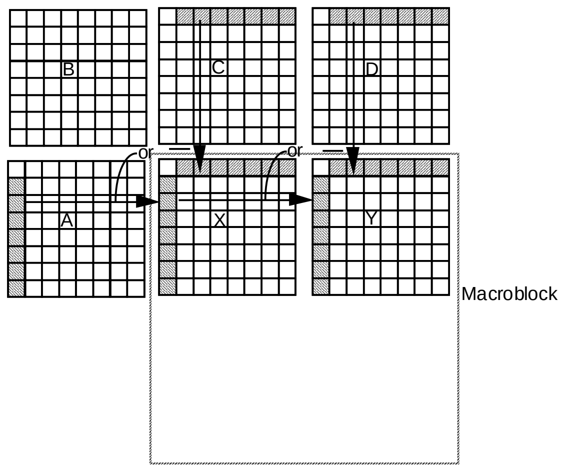


Figure 15 Prediction of AC coefficients of blocks in an intra macroblock

The predicted DC and AC coefficients (as well as the unpredicted AC coefficients) of DCT blocks of intra macroblocks are scanned by one of the three scans [38]: alternate-horizontal, alternate-vertical (MPEG-2 interlace scan) and the zigzag scan (normal scan used in H.263 and MPEG-1). The actual scan used depends on the coefficient predictions used. For instance, if AC coefficient prediction is disabled for a intra macroblock, all blocks in that macroblock are zigzag-scanned. If AC coefficient prediction is enabled and DC coefficient prediction was selected from the horizontally adjacent block, alternate-vertical scan is used, likewise, if AC coefficientprediction is enabled and DC coefficient prediction was selected from the vertically adjacent block, alternate-horizontal scan isused. Figure 16 shows the three scans used.

0	1	2	3	10	11	12	13
4	5	8	9	17	16	15	14
6	7	19	18	26	27	28	29
20	21	24	25	30	31	32	33
22	23	34	35	42	43	44	45
36	37	40	41	46	47	48	49
38	39	50	51	56	57	58	59
52	53	54	55	60	61	62	63

(a)

0	4	6	20	22	36	38	52
1	5	7	21	23	37	39	53
2	8	19	24	34	40	50	54
3	9	18	25	35	41	51	55
10	17	26	30	42	46	56	60
11	16	27	31	43	47	57	61
12	15	28	32	44	48	58	62
13	14	29	33	45	49	59	63

(b)

0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

(c)

Figure 16 Scans for intra blocks: (a) Alternate-horizontal (b) Alternate-vertical (c) Zig-zag

A three dimensional variable length code is used to code the scanned DCT events of intra blocks. An event is a combination of three items (last, run, level). The 'last' indicates if a coefficient is the last nonzero coefficient of a block or not, the 'run' indicates number of zero coefficients preceeding the current nonzero coefficient and level indicates the amplitude of the quantized coefficient.

The DCT coefficients of inter blocks, unlike DCT coefficients of inter blocks do not undergo any prediction or adaptive scanning, in fact they use the fixed zigzag scan of Figure 16(c). The scanned coefficients of inter blocks are also coded by a three dimensional variable length code table with similar structure as the intra variable length code table but with code entries optimized for inter statistics.

Finally, as mentioned earlier, MPEG-4 also supports efficient coding of interlaced video. It combines the macroblock based frame/field DCT coding of MPEG-2 with the improved DC coefficient coding, quantization, scanning and variable length coding of normal MPEG-4 video coding resulting in improved coding efficiency. Furthermore, it allows DCT coding of arbitrary shaped VOPs of interlaced video where as MPEG-2 only supports rectangular pictures of interlaced video.

4.2.4 Sprite Coding

In computer games, a sprite refers to an synthetic object that undergoes some form of transformation (including animation). Also in literature, in connection with highly efficient representation of natural video, the term 'mosaic' or 'world image' is used to describe a large image built by integration of many frames of a sequence spatially and/or many frames of a sequence temporally; in MPEG-4 terminology, such an image is referred to as a static sprite. Static sprites can improve the overall coding efficiency, for example, by coding the background only once and warping it to generate the rendition required at a specific time instance.

A static sprite [26,37] is usually built offline and can be used to represent synthetic or natural objects. It is quite suitable for natural objects that undergo rigid motion and where a wall paper like rendering is sufficient. One of the main components in coding using natural sprites is generation of the sprite itself. For generating a static sprite, the entire video object is assumed to be available. For generating a static sprite, the entire video object is assumed to be available. For each VOP in the VO, the global motion is estimated according to a transformation model (say, perspective transformation) using which a VOP is then registered with the sprite by warping the VOP to sprite coordinate system. Finally, the warped VOP is blended with the sprite which is used for estimation of motion of the subsequent VOP.

A number of choices regarding the transformations models exist such as stationary, translation, magnification-rotation-translation, affine, and perspective transformation. Each transformation can be defined as either a set of coefficients or the motion trajectories of some reference points; the former, is convenient for performing the transformations whereas the later for encoding the transformations. If four reference points are used, perspective transformation can be employed for warping and is defined by the following.

$$x' = (ax + by + c) / (gx + hy + l)$$

$$y' = (dx + ey + f) / (gx + hy + l)$$

where {a, b, c, d, e, f, g, h, l } are the coefficients of the transformation, (x,y) is one of the reference points of interest in current VOP which corresponds to point (x',y') in the sprite, expressed in sprite coordinate system.

Once the sprite is available, global motion between the current VOP and the sprite is estimated, using the perspective transform, for example. The reconstructed VOPs are generated from the sprite by directly warping the quantized sprite using specified motion parameters. Residual error between the original VOP and the warped sprite is not sent.

4.3 Scalable Video Coding

Scalability of video is the property that allows a video decoder to decode portions of the coded bitstreams to generate decoded video of quality commensurate with the amount of data decoded. In other words, scalability allows a simple video decoder to decode and produce basic quality video while an enhanced decoder may decode and produce enhanced quality video, all from the same coded video bitstream. This is possible because scalable video encoding ensures that input video data is coded as two or more layers, an independently coded base layer and one or more enhancement layers coded dependently, thus producing scalable video bitstreams. The first enhancement layer is coded with respect to the base layer, the second enhancement layer with respect to the first enhancement layer and so forth.

Scalable coding offers a means of scaling the decoder complexity if processor and/or memory resources are limited and often time varying. Further, scalability also allows graceful degradation of quality when the bandwidth resources are also limited and continually changing. It also allows increased resilience from errors under noisy channel conditions.

MPEG-4 offers a generalized scalability [26,37,38] framework supporting both the Temporal and the Spatial scalabilities, the primary type of scalabilities. Temporally scalable encoding offers decoders a means to increase temporal resolution of decoded video using decoded enhancement layer VOPs in conjunction with decoded base layer VOPs. Spatial scalability encoding on the other hand offers decoders a means to decode and display either the base layer or the enhancement layer output; typically, since base layer uses one-quarter resolution of the enhancement layer, the enhancement layer output provides the better quality, albeit requiring increased decoding complexity. The MPEG-4 generalized scalability framework employs modified B-VOPs that only exist in enhancement layer to achieve both temporal and spatial scalability; the modified enhancement layer B-VOPs use the same syntax as normal B-VOPs but for modified semantics which allows them to utilize a number of interlayer prediction structures needed for scalable coding

Figure 17 shows an example of the prediction structure used in temporally scalable coding. The base layer is shown to have one-half of the total temporal resolution to be coded, the remaining one-half is carried by the enhancement layer. Base layer is coded independently as in normal video coding where as the enhancement layer uses B-VOPs that use both, an immediate temporally previous decoded base layer VOP as well as an immediate temporally following decoded base layer VOP for prediction.

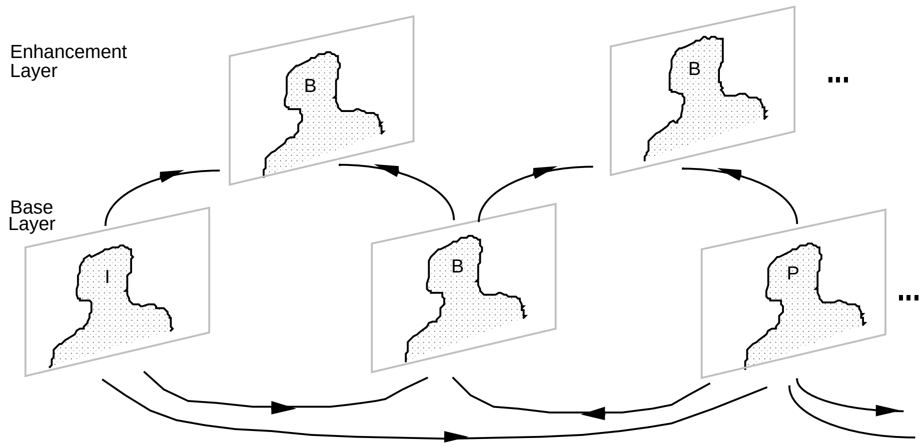


Figure 17 Temporal Scalability

Next, Figure 18 shows an example of prediction structure used in spatially scalable coding. The base layer is shown to have one-quarter resolution of the enhancement layer. Base layer is coded independently as in normal video coding where as the enhancement layer mainly uses B-VOPs that use both, an immediate previous decoded enhancement layer VOP as well as a coincident decoded base layer VOP for prediction.

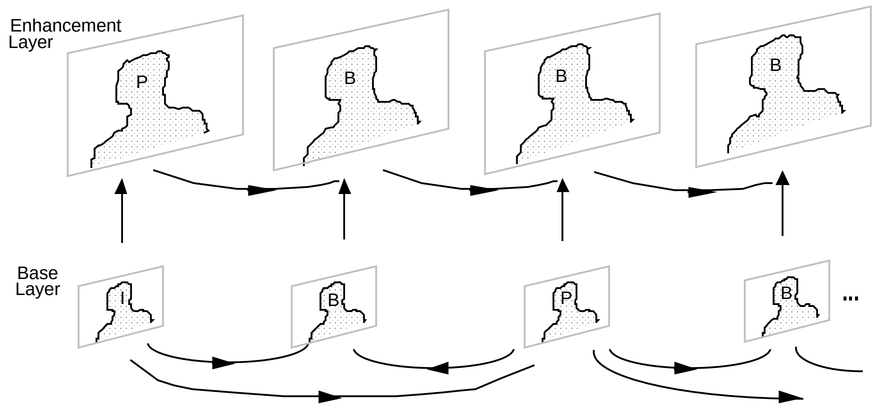


Figure 18 Spatial scalability

In reality, some flexibility is allowed in choice of spatial and temporal resolutions for base and enhancement layers as well as the prediction structures allowed for the enhancement layer to cope with a variety of conditions in which scalable coding may be needed. Further, both spatial and temporal scalability with rectangular VOPs and temporal scalability of arbitrary shape VOPs is expected to be supported in MPEG-4 Version 1. Figure 17 and 18 are applicable not only to rectangular VOP scalability but also to arbitrary shape VOP scalability (in this case only the shaded region depicting the head and shoulder view is used for predictions in scalable coding, and the rectangle represents the bounding box).

4.4 Robust Video Coding

Truly robust video coding requires a diversity of strategies. MPEG-4 video offers a number of tools which an encoder operating in the error resilient mode [37] can employ. MPEG-4 video also offers other tools (not specific to error resilience) that can be used to provide robust video coding. We now discuss the various available tools and how they can be used by themselves or in conjunction to provide robust video coding.

4.4.1 Object Priorities

The object and based organization of MPEG-4 video potentially makes it easier to achieve a higher degree of error robustness due to the possibility of prioritizing each semantic object based on its relevance. Further, the MPEG-4 systems, since it offers scene description and composition flexibilities can ensure that the reconstructed scenes are meaningful even if low priority objects are only partially available or become unavailable (say due to data loss or corruption). Currently, the MPEG-4 systems offers a way of providing priorities to each stream, if these are found insufficient, priorities may also be assigned to VOs and VOLs in video bitstream; this has not yet been resolved. Further, VOP types themselves lend to a form of automatic prioritization since, B-VOPs are noncausal and do not contribute to error propagation and thus can be assigned a lower priority and perhaps even be discarded in case of severe errors.

Although in principle, coding of scenes as arbitrary shaped objects can be advantageous, the down side can be shape overhead, increase in decoding complexity and the sensitivity of shape coding (which can be interframe and context dependent) to errors.

4.4.2 Resynchronization

VOPs already offer a means of resynchronization to prevent accumulation of errors. Further, it is possible for encoder to offer increased error resilience by placing resynchronization (resync) marker in the bitstream with approximately constant spacing. The resync marker is a unique 17 bit code that normally can not be emulated by any valid combination of VLC codewords that may precede it. The VM7 error resilient encoding recommends the spacings of Table 5, in bits as a function of the coding bitrate.

Table 5 Recommended spacings for resync markers

Bitrate (kbit/s)	**Spacing (bits)**
<= 24	480
25 - 48	736

In fact, to enable recovery from errors, a video packet header is used which in addition to resync marker contains macroblock number, quantizer scale and extension header (optional, when present, it includes vop time and coding type information), the timing information ensures that decoder can determine the VOP to which the video packet header belongs.

4.4.3 Data Partitioning

Data partitioning allows a mechanism to provide increased error resilience by separating the normal motion and texture data of all macroblocks in a video packet and sending all of the motion data followed by a motion marker followed by all of the texture data. The motion marker is a unique 17 bit code that can not be emulated by any valid combination of VLC codewords that may precede it.

The motion data per macroblock is arranged to contain coded/not coded information followed by combined macroblock type and coded block pattern information followed by motion vector(s); the motion data of the next macroblock follows that of the previous macroblock till the motion data for all macroblocks in the video packet can be sent. The texture data per macroblock is arranged in two parts. The first part contains coded block information of luminance blocks in a macroblock followed by optional differential quantizer information, this is repeated for all macrolocks in the video packet. The second part contains coded DCT coefficients of a macroblock, followed by that of the next macroblock till DCT coefficients for all macroblocks in the video packet can be sent.

4.4.4 Reversible VLCs

The reversible VLCs offer a mechanism for decoder to recover additional texture data in the presence of errors since the special design of reversible VLCs enables decoding of codewords in both the forward (normal) and the reverse direction. The encoder decides whether for coding of DCT coefficients, to use the reversible VLCs or normal VLCs (depending on the coding efficiency versus error resilience tradeoffs needed) by signalling this information as part of the bitstream. It is possible to invoke the error resilience mode independent of whether reversible VLCs are used or not.

The process of additional texture recovery in a corrupted bitstream starts by first detecting the error and searching forward in the bitstream to locate the next resync marker. Once the next resync marker is located, from that point, due to use of reversible VLCs for texture coding, the texture data can be decoded in the reverse direction until an error is detected. Further, when errors are detected in texture data, the decoder can use correctly decoded motion vector information to perform motion compensation and conceal these errors.

4.4.5 Intra Update

Typically, intra coding of macroblocks although it can provide refresh from coding errors, is expensive when used very frequently due to higher coding cost when video data is coded in intra mode. In MPEG-4, considerable effort has been placed in improving the efficiency of intra coding and the resulting scheme offers higher efficiency than H.263 or MPEG-1 based intra coding. Thus, encoders requiring higher error resilience can choose to code increased number of macroblocks in intra coding mode than with previous standards, providing an improved refresh from coding errors.

4.4.6 Scalability for Robustness

Scalable coding can offer a means of graceful degradation in quality when packet errors due to noisy conditions or packet losses due to congestion on the network are likely. Since scalable coding involves independent coding of the base layer, the base layer data can be assigned a higher priority and be better protected. Since, the enhancement layers only offer improvement in spatial or temporal resolution, the enhancement layer data can be assigned a lower priority.

In discussing scalability and the performance tradeoffs/benefits it offers, it is important to distinguish that while spatial scalability of video usually incurs some performance degradation and increase in complexity, temporal scalability of video on the other hand does not involve any degradation of performance or increase in complexity. Further, even for spatial scalability, the loss depends on many factors such as the choice of resolutions in the base and the enhancement layer(s), downsampling filter, prediction configuration, complexity of scenes etc, and can be minimized with some care. Thus, in particularly noisy conditions, selective use of scalability can increase robustness significantly with very little degradation in performance or increase in complexity.

4.4.7 Correction and Concealment

Due to the channel specific nature of the degree and type of error correction needhed, MPEG-4 is not likely to recommend a specific error correction method, but leaves it up to the chosen data transport layer to implement the needed technique. Further, error concealment strategies although encouraged are not standardized by MPEG-4; perhaps the work done on this topic in MPEG-2 can be useful.

4.5 Facial Animation Coding

The Facial animation Parameters (FAPs) and the Facial Definition Parameters (FDPs) [26] are sets of parameters designed to allow animation of faces reproducing expressions, emotions and speech pronunciation, as well as, definition of facial shape and texture. The same set of FAPs when applied to different facial models result in reasonably similar expressions and speech pronunciation without the need to initialize or calibrate the model. The FDPs, on the other hand, allow the definition of a precise facial shape and texture in the setup phase. If the FDPs are used in the setup phase, it is also possible to prrecisely produce the movements of particular facial features. Using a phoneme to FAP conversion it is possible to control facial models accepting FAPs via text to speech (TTS) systems; this conversion is not standardized. Since, it is assumed that every decoder has a default face model with default parameters the set up stage is not necessary to create face animation but for customizing the face at the decoder.

The FAP set contains two high level parameters *visemes* and *expressions*. A viseme is a visual correlate to a phoneme. The viseme parameter allows viseme rendering (without having to express them in terms of other parameters) and enhances the result of other parameters, insuring the correct rendering of visemes. Only static visemes which are clearly distinguished are included in the standard set; examples of suchviseme are shown in Table 6. The expression parameter similarly allows definition of high level facial expressions. The facial expression parameter values are defined by textual descriptions such as, joy, sadness, anger, fear, disgust, surprise.

Table 6 Viseme number 1 to 14, its related phoneme set and examples

1	2	3	4	5	6	7	8	9	10	11	12	13	14
(p,b, m)	(f,v)	(T,D)	(t,d)	(k,g)	(tS,d Z,S)	(s,z)	(n,l)	(r)	(A:)	(e)	(I)	(Q)	(U)
put	far	think	tip	call	chair	sir	lot	red	car	bed	tip	top	book
bed	voice	that	doll	gas	join	zeal	not						
mill					she								

All the parameters involving translational movement are expressed in terms of the Facial Animation Parameter Units (FAPU). These units are defined in order to allow interpretation of the FAPs on any facial model in a consistent way, producing reasonable results in terms of expression and speech pronunciation. The measurement units are shown in Figure 19 and are defined as follows.

IRISD0 = Iris diameter (by definition it is equal to the distance between upper and lower eyelid) in neutral face; IRISD = IRISD0 / 1024

ES0 = Eye separation; ES = ES0 / 1024

ENS0 = Eye - nose separation; ENS = ENS0 / 1024

MNS0 = Mouth - nose separation; MNS = MNS0 / 1024

MW0 = Mouth width; MW = MW0 / 1024

AU = Angle Unit = 10E-5 rad

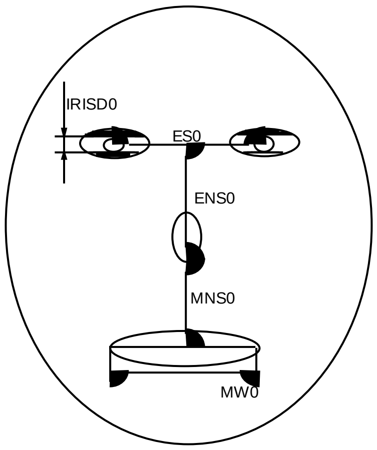


Figure 19 Facial animation parameter units

The FDPs are used to customize the proprietary face model of the decoder to a particular face or to download a face model along with the information about how to animate it. The FDPs are normally transmitted once per session, followed by a stream of compressed FAPs. However, if the decoder does not receive the FDPs, the use of FAPUs ensures that it can still interpret the FAP stream. This ensures minimal operation in broadcast or teleconferencing applications.

The FDP set is specified using FDP node (in MPEG-4 systems) which defines the face model to be used at the receiver. Two options are supported:

- Calibration information is downloaded so that the proprietary face of the receiver can be configured using facial feature points and optionally a 3D mesh or texture.
- A face model is downloaded with the animation definition of the Facial Animation Parameters. This face model replaces the proprietary face model in the receiver.

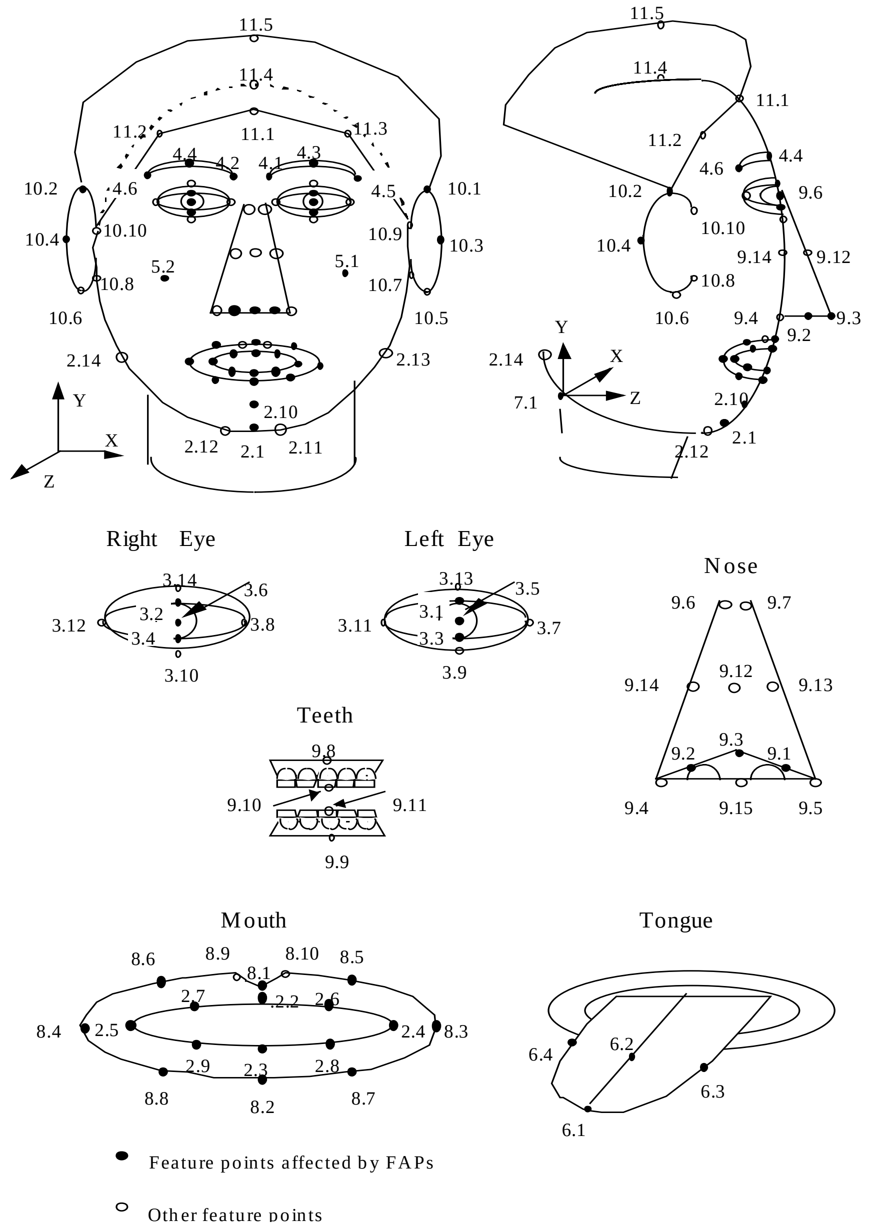


Figure 20 Facial definition feature set

4.6 Object Mesh Coding

For general natural or synthetic visual objects, mesh based representation [26] can be useful for enabling a number of functions such temporal rate conversion, content manipulation, animation, augmentation (overlay), transfiguration (merging or replacing natural video with synthetic) and others. MPEG-4 includes a tool for triangular mesh based representation of general purpose objects.

A visual object of interest, when it first appears (as a 2D VOP) in the scene, is tassellated into triangular patches resulting in a 2D triangular mesh. The vertices of the triangular patches forming the mesh are referred to as the *node points*. The node points of the initial mesh are then tracked as the VOP moves within the scene. The 2D motion of a Video Object can thus be compactly represented by the motion vectors of the node points in the mesh. Motion compensation can then be achieved by texture mapping the patches from VOP to VOP according to affine transforms. Coding of video texture or still texture (to be discussed next) of object is performed by the normal texture coding tools of MPEG-4. Thus, efficient storage and transmission of the mesh representation of a moving object (dynamic mesh) requires compression of its geometry and motion.

The initial 2D triangular mesh is either a uniform mesh or a Delaunay mesh, the mesh triangular topology (links between node points) is not coded; only the 2D node point coordinates $\vec{p}_n = (x_n, y_n)$ are coded. A uniform mesh can be completely specified using five parameters such as the number of nodes horizontally and the number of nodes vertically, the horizontal and the vertical dimensions of each quadrangle consisting of two triangles, and the type of splitting applied on each quadrangle to obtain triangles. For a Delaunay mesh, the node point coordinates are coded by first coding the boundary node points and then the interior node points of the mesh. To encode the interior node positions, the nodes are traversed one by one using a *nearest neighbor* strategy. A linear ordering of the node points is computed such that each node is visited only once. When a node is visited, its position is differentially coded with respect to the position of previous coded node used as the predictor. By sending the total number of node points and the number of boundary node points, the decoder knows how many node points will follow, and how many of those are boundary nodes; thus it is able to reconstruct the polygonal boundary and the locations of all nodes. The mesh based representation of an object and the traversal of nodes for mesh geometry coding is illustrated in Figure 21 by an example.

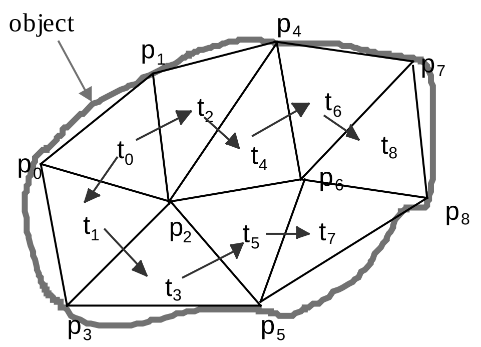


Figure 21 2D Mesh representation of an object, and coding of mesh geometry

First, the total number of nodes and the number of boundary nodes is encoded. The top-left node $\vec{p}_0$ is coded without prediction. Then, the next clockwise boundary node $\vec{p}_1$ is found and the difference between $\vec{p}_0$ and $\vec{p}_1$ is encoded; then all other boundary nodes are encoded in a similar fashion. Then, the not previously encoded interior node that is nearest to the last boundary node is found and the difference between these is encoded; this process is repeated until all the interior nodes are covered. The mesh geometry is only encoded when a new mesh needs to be initialized with respect to a particular VOP of the corresponding visual object; it consists of the initial positions of the mesh nodes.

The mesh motion is encoded [39] at subsequent time instants to describe the motion of the corresponding video object; it consists of a motion vector for each mesh node such that the motion vector points from a node point of the previous mesh in the sequence to a node point of the current mesh.

The mesh bitstream syntax consists of the two parts: mesh geometry and mesh motion. The node coordinates and node motion vectors are specified to one-half pixel accuracy.

4.7 Still Texture Coding

The Discrete Wavelet Transform (DWT) [26,37] is used to code still image data employed for texture mapping. Besides coding efficiency, an important requirement for coding texture map data is that it should be coded in a manner facilitating continuous scalability, thus allowing many resolution/qualities to be derived from the same coded bitstream. While DCT based coding is able to provide comparable coding efficiency as well as a few scalability layers, DWT based coding offers flexibility in organization and number of scalability layers.

The principle of DWT encoding is shown in Figure 22.

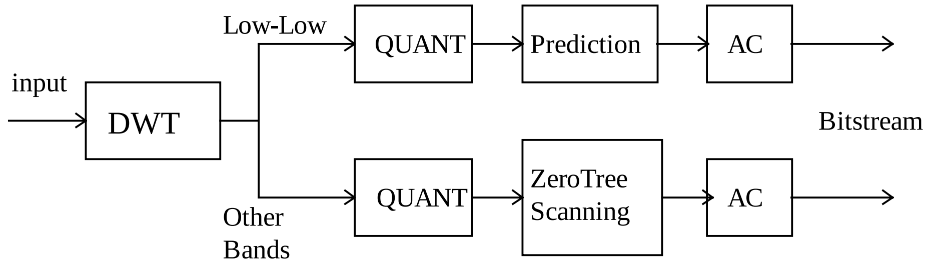


Figure 22 Block diagram of the DWT encoder of still image texture

The basic modules of a zero-tree wavelet based coding scheme are as follows:

1. Decomposition of the texture using discrete wavelet transform (DWT).
2. Quantization of the wavelet coefficients.
3. Coding of the lowest frequency subband using a predictive scheme.
4. Zero-tree scanning of the higher order subband wavelet coefficients.
5. Entropy coding of the scanned quantized wavelet coefficients and the significance map.

A 2d separable wavelet decomposition is applied to the still texture to be coded. The wavelet decomposition is performed using a Daubechies (9,3) tap biorthogonal filter which has been shown to provide good compression performance. The filter coefficients are:

Lowpass = [0.03314563036812, -0.06629126073624, -0.17677669529665,
0.41984465132952, 0.99436891104360, 0.41984465132952,
-0.17677669529665, -0.06629126073624, 0.03314563036812]

Highpass = [-0.35355339059327, 0.70710678118655, -0.35355339059327]

A group delay is applied to each filter to avoid the phase shift on both of the image domain and the wavelet domain.

The wavelet coefficients of the lowest band are coded independently from the other bands. These coefficients are quantized using an uniform midrise quantizer. After quantization of the lowest subband coefficients, an implicit prediction (same as that used for DC prediction in intra DCT coding) is applied to compute prediction error which is then encoded using an adaptive arithmetic coder which uses min-max coding.

The wavelet coefficients of the higher bands are first quantized by multilevel quantization which provides the flexibility needed to tradeoff number of levels, type of scalability (spatial or SNR), complexity and coding efficiency. Different quantization step sizes (one for luminance and one for chrominance) can be specified for each level of scalability. All the quantizers of the higher bands are uniform mid-rise quantizer with a dead zone 2 times the quantization step size. The quantization step sizes are specified by the encoder in the bitstream. In order to achieve the fine granularity SNR scalability, bi-level quantization scheme is used for all the multiple quantizers. This quantizer is also a uniform mid rise quantizer with a dead zone 2 times the quantization step size. The coefficients that lie outside the dead zone (in the current and previous pass) are quantized with a 1 bit accuracy. The number of quantizers is equal to the maximum number of bitplanes in the wavelet transform representation. In this bi-level case, instead of the quantization step sizes, the maximum number of the bitplanes is specified in the bitstream.

After quantization, each wavelet coefficient is either zero or nonzero. The coefficients of all bands (except the lowest) are scanned by zero-tree scanning. Zero-tree scanning is based on the observation that strong correlation exists in the amplitudes of the wavelet coefficients across scales, and on the idea of partial ordering of the coefficients. The coefficient at the coarse scale is called the parent, and all coefficients at the same spatial location, and of similar orientation, at the next finer scale are that parent's children. Since the lowest frequency subband is coded separately, the wavelet trees start from the adjacent higher bands. In order to achieve a wide range of scalability levels efficiently as needed by the application, a multiscale zerotree coding scheme is employed. The zero-tree symbols and the quantized values are coded using an adaptive arithmetic encoder which uses a three-symbol alphabet.

The process of scalable decoding the various spatial/SNR layers from a single DWT coded bitstream is shown in Figure 23.

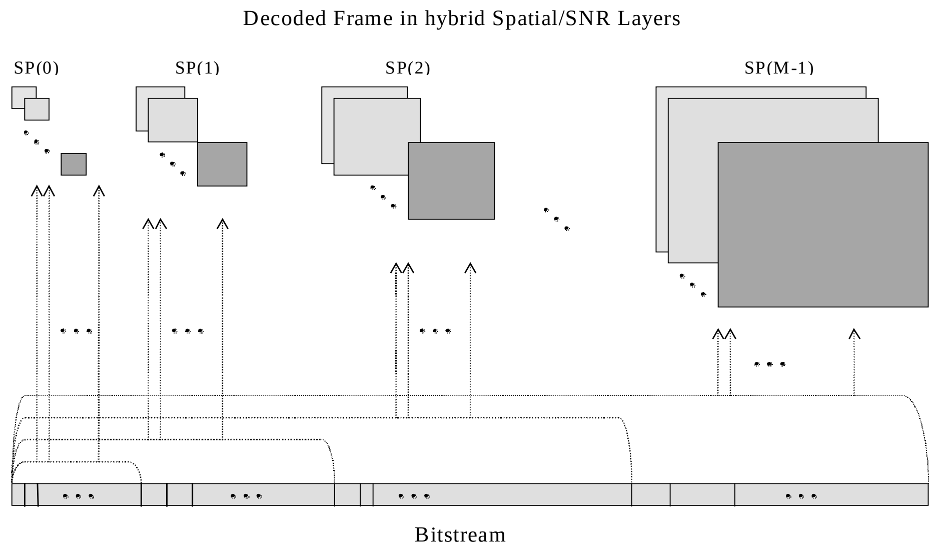


Figure 23 Scalable decoding of still object texture

4.8 View Dependent Texture Coding

View dependent texture coding [26] is designed for incrementally streaming the texture data when the viewpoint is moving. This technique is intended for streaming texture across the network allowing low-bandwidth communication of remote virtual environments when very low delay transmission is possible. The streaming of texture data id done using a back-channel from the decoder to coder. The back channel is used to indicate to the coder the current viewing conditions and thus it becomes possible to send from the coder to the decoder only the data that is really needed.

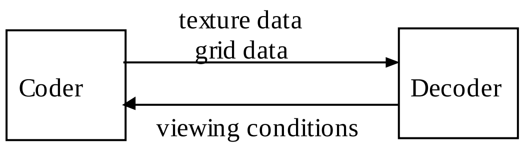


Figure 24 View Dependent texture system

The two main stages in transmission are the initialization and the streaming stages.

The basic operations that are performed by the coder are as follows. At the encoder, image to be coded is wavelet transformed, its useful coefficients are selected and coded and transmitted. At the decoder, the received coefficients are decoded and inverse transformed. The resulting image is mapped on to a mesh grid using orthographic or perspective projection.

In orthographic projection, at the decoder side, the knowledge of the 3D regular mesh grid is needed on which texture is mapped. The texture mapping operation is defined using the coordinate vertices of the mesh grid. In perspective projection, the same principle is used. In order to take into account perspective distortion, the resolution of texture image is increased as the distance of the viewer and the cell center is decreased.

Feedback is sent to encoder regarding the viewpoint and only the encoder send only the necessary texture data.

5. MPEG-4 Audio

The benefits of MPEG-4 speech coder can be exploited in a number of applications. As an example, the MPEG-4-based internet-phone system offers robustness against packet loss or change in transmission bitrates. When applied in an audio-visual system, the coding quality is improved by assigning the audio and the visual bitrates adaptively based on the audio-visual content. As an another example, the MPEG-4 speech coder could also be used in radio communication systems where it can offer higher error robustness by changing the bit-allocation between speech coding and error correction depending on the error conditions. These features have not been realized by any other standard. Furthermore, low bitrate is useful for "Party talk". Although up to 10 persons can have conversation simultaneously over the Internet, one terminal only has to receive bitrate of only 18 kbit/s, to hear the conversation of 9 other persons.

Besides speech coding MPEG-4 also offers multichannel audio coding based on optimized MPEG-2 AAC coding. MPEG-4 also offers solutions for medium bitrate audio coding. Furthermore, it supports the concept of audio objects. Just as video scenes are made from visual objects, audio scenes may be usefully described as the spatiotemporal combination of audio objects. An “audio object” is a single audio stream coded using one of the MPEG-4 coding tools, like CELP or Structured Audio. Audio objects are related to each other by mixing, effects processing, switching, and delaying them, and may be spatialized to a particular 3-D location. The effects processing is described abstractly in terms of a signal-processing language (the same language used for Structured Audio), so content providers may design their own empirically, and include them in the bitstream.

5.1 Natural Audio

As mentioned earlier, Natural Audio coding [23,27] in MPEG-4 consists of the following.

- The lowest bitrate range between 2 and 6 kbit/s is covered by *Parametric Coding* (mostly for speech coding)
- The medium bitrates between 6 and 24 kbit/s use *Code Excited Linear Predictive (CELP)* with two sampling rates, 8 and 16 kHz, for a broader range of audio signals
- For higher bitrates starting at about 16 kbits/s, frequency domain coding techniques are applied, e.g., optimized version of MPEG-2 *Advanced Audio Coding* (AAC). The audio signals in this region typically have bandwidths starting at 8 kHz.

Figure 25 provides a composite picture of the applications of MPEG-4 audio and speech coding, the signal bandwidth and the type of coders used.

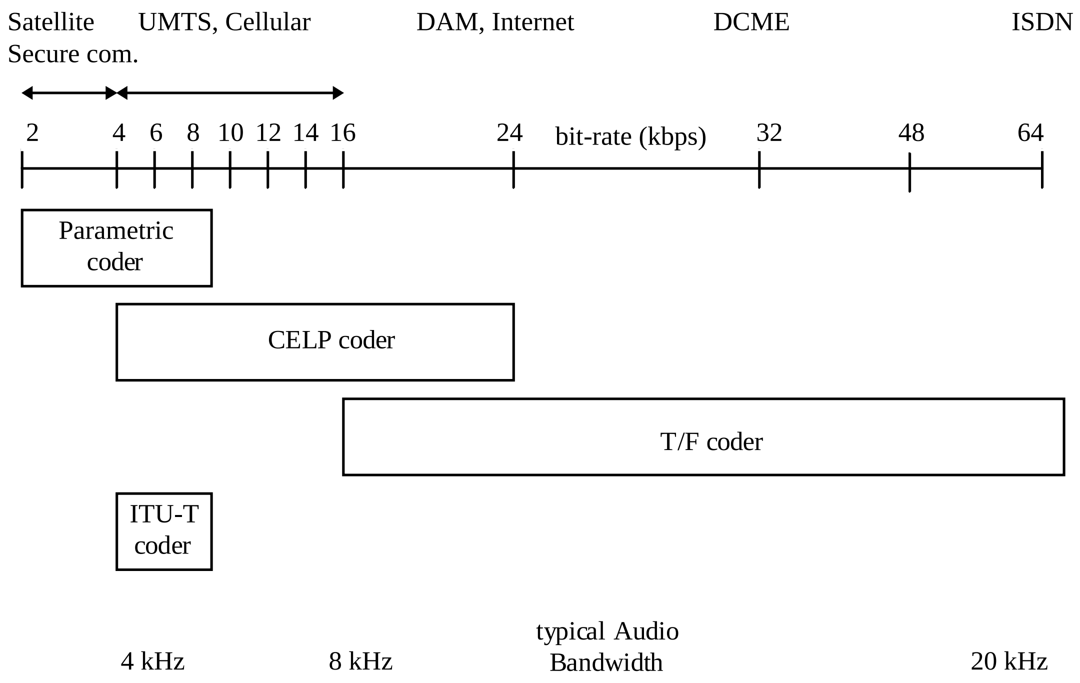


Figure 25 Natural Audio Coding in MPEG-4

5.1.1 Parameteric Coder

The parametric coder core provides two sets of tools. The HVXC coding tools (Harmonic Vector eXcitation Coding) allow coding of speech signals at 2 kbit/s while the Individual Line coding tools allow coding of non-speech signals like music at bit rates of 4 kbit/s and higher. Both sets of tools allow independent change of speed and pitch during the decoding and can be combined to handle a wider range of signals and bit rates.

We now list the encoder and the decoder tools in HVXC and Individual Line coding [27]; however, only the decoder tools are normative.

HVXC Encoder Tools	*HVXC Decoder Tools*
1. Normalization	1. LSP Decoder
2. Pitch Estimation	2. Harmonic Decoder

3. Harmonic Magnitude Extraction	3. Time Domain Decoder
4. Perceptual Weighting	4. Speed Control
5. Harmonic Encoding	5. Harmonic Synthesizer
6. V/UV Decision	6. Time Domain Synthesizer
7. Time Domain Coder	7. Postprocessing
8. Short Frame Mode	8. Short Frame Mode

Individual Line Encoder Tools	*Individual Line Decoder Tools*
1. Individual Line Parameter Extraction	1. Individual Line Bitstream Decoding
2. Individual Line Bitstream Encoding	2. Individual Line Synthesizer

5.1.2 CELP Coder

CELP coder is designed for speech coding at two different sampling frequencies, namely, 8 kHz and 16 kHz. The speech coders using 8 kHz sampling rate are referred to as narrowband coders while those using 16 kHz sampling rate are wideband coders. CELP coder includes tools offering a variety of functions including bit rate control, bit rate scalability, speed control, complexity scalability and speech enhancement. Using the narrowband and the wideband CELP coders, it is possible to span a wide range of bit rates (4 kbps to 24 kbps). Real-time bit-rate control in small steps can be provided. A common structure of tools have been defined for both the narrowband and wideband coders; many tools and processes have been designed to be commonly usable for both narrowband and wideband speech coders.

We now list the encoder and the decoder tools in CELP coding [27]; however, only the decoder tools are normative.

CELP Encoder Tools	*CELP Decoder Tools*
1. Preprocessing	1. Bitstream Demultiplexer
2. LPC Analysis	2. LPC Decoder
3. LPC Quantizer	3. Excitation Generator
4. Perceptual Weighting	4. LPC Synthesis Filter
5. Pitch Estimation	5. LPC Postprocessing
6. Analysis Filter	6. Weighting Module
7. Weighting Module	
8. Bitstream Multiplexer	

5.1.3 Time/Frequency Coder

The high-end audio coding in MPEG-4 [27] is based on MPEG-2 AAC coding. MPEG-2 AAC is a state-of-the-art audio compression algorithm that provides compression superior to that provided by older algorithms. AAC is a transform coder and uses a filterbank with a finer frequency resolution that enables superior signal compression. AAC also uses a number of new tools such as temporal noise shaping, backward adaptive linear prediction, joint stereo coding techniques and Huffman coding of quantized components, each of which provide additional audio compression capability. Furthermore, AAC supports a wide range of sampling rates and bitrates, from one to 48 audio channels, up to 15 low frequency enhancement channels, multi-language capability and up to 15 embedded data streams. MPEG-2 AAC provides a 5-channel audio coding capability, while being a factor of two better in coding efficiency relative to MPEG-2 BC; since AAC has no such backward compatibility requirement and it incorporates the recent advances, in MPEG formal listening tests for 5-channel audio signals, it provided slightly better audio quality at 320 kbit/s than MPEG-2 BC can provide at 640 kbit/s.

5.2 Text to Speech

Text-to-Speech (TTS) conversion system synthesizes speech as its output when a text is accessed as its input. In other words, when the text is accessed, the TTS changes the text into a string of phonetic symbols and the corresponding basic synthetic units are retrieved from the pre-prepared database. And then the TTS concatenates the synthetic units to synthesize the output speech with the rule-generated prosody. Some application areas for MPEG-4 TTS are as follows.

- Artificial Story Teller. (or Story Teller on Demand)
- Synthesized speech output synchronized with Facial Animation (FA).
- Speech synthesizer for avatars in various Virtual Reality (VR) applications.
- Voice News Paper.
- Dubbing tools for animated pictures.
- Voice Internet
- Transportation Timetables

The MPEG-4 TTS [27] can not only synthesize speech according to the input speech with a rule-generated prosody, but also executes several other functions. They are as follows.

1) speech synthesis with the original prosody from the original speech,
2) synchronized speech synthesis with Facial Animation (FA) tools,
3) synchronized dubbing with moving pictures not by recorded sound but by text and some lip shape information,
4) trick mode functions such as stop, resume, forward, backward without breaking the prosody even in the applications with Facial Animation (FA)/ Motion Pictures (MP),
5) users can change the replaying speed, tone, volume, speaker's sex, and age.

MPEG-4 TTS can be used for many languages in the world since it adopts the concept of the language code such as the country code for an international call. Presently, only 25 countries, i.e., the current ISO members, have their own code numbers, to identify that their own language has to be synthesized, except the International Phonetic Alphabet (IPA) code assigned as 0. However, 8 bits have been assigned for the language code to ensure that all countries can be assigned language code when asked in future. IPA could be used to transmit all languages.

For MPEG-4 TTS, only the interface bitstream profiles are the subject of standardization. Because there are already many different types of TTS and each country has several or a few tens of different TTSs synthesizing its own language, it is impossible to standardize all the things related to TTS. However, it is believed that almost all TTSs can be modified to accept MPEG-4 TTS interface very quickly by a TTS expert because of the rather simple structure of the MPEG-4 TTS interface bitstream profiles.

5.3 Structured Audio

Structured audio formats use ultra-low bit-rate algorithmic sound models to code and transmit sound. MPEG-4 standardizes an algorithmic sound language and several related tools for the structured coding of audio objects. Using these tools, algorithms which represent the exact specification of a sound scene are created by the content designer, transmitted over a channel, and executed to produce sound at the terminal. Structured audio techniques in MPEG-4 [27] allow the transmission of synthetic music and sound effects at bit-rates from 0.01 to 10 kbit/s, and the concise description of parametric sound post-production for mixing multiple streams and adding effects processing to audio scenes

MPEG-4 does not standardize a synthesis method, but a signal-processing language for describing synthesis methods. SAOL, pronounced “sail”, stands for “Structured Audio Orchestra Language” and is the signal-processing language enabling music-synthesis and effects post-production in MPEG-4. It falls into the music-synthesis category of “Music V” languages; that is, its fundamental processing model is based on the interaction of oscillators running at various rates. However, SAOL has added many new capabilities to the Music V language model which allow for more powerful and flexible synthesis description. Using this language, any current or future synthesis method may be described by a content provider and included in the bitstream. This language is entirely normative and standardized, so that every piece of synthetic music will sound exactly the same on every compliant MPEG-4 decoder, which is an improvement over the great variety in MIDI-based synthesis systems

The techniques required for automatically producing a Structured Audio bitstream from an arbitrary sound are beyond today’s state of the art, although they are an active research topic. These techniques are often called “automatic source separation” or “automatic transcription”; there are many references within the audio processing literature on the capabilities of today’s methods. In the mean time, content authors will use special content creation tools to directly create Structured Audio bitstreams. This is not a fundamental obstacle to the use of MPEG-4 Structured Audio, because these tools are very similar to the ones that content authors use already; all that is required is to make them capable of producing MPEG-4 output bitstreams.

There is no fixed complexity which is adequate for decoding every conceivable Structured Audio bitstream. Simple synthesis methods are very low-complexity, and complex synthesis methods require more computing power and memory. Since the description of the synthesis methods is under the control of the content provider, the content provider is responsible for understanding the complexity needs of his bitstreams. Past versions of structured audio systems with similar capability have been optimized to provide multitimbral, highly polyphonic music and post-production effects in real-time on a 150 MHz Pentium computer or simple DSP chip. One “level” of capability in the Synthetic Object Audio profile of MPEG-4 Audio may provide for simpler and/or less normative synthesis methods in RAM- or computing-limited terminals.

6. MPEG-4 SYSTEMS

The Systems [25] part of MPEG-4 perhaps represent the most radical departure from previous MPEG specifications. Indeed, the object-based nature of MPEG-4 necessitates a totally new approach on what the Systems layer is required to provide. Issues of synchronization and multiplexing are, of course, still very essential. Note, however, that an MPEG-4 scene may be composed of several objects, and hence synchronization between a large number of streams is required. In addition, the spatio-temporal positioning of such objects forming a scene (or scene description) is a key component. Finally, issues of interactivity are also quite new in MPEG as well.

6.1 System Decoder Model

A key problem in designing an audiovisual communication system is ensuring the time is properly represented and reconstructed by the terminal. This serves two purpose: 1) it ensures that “events” occur at designated times as indicated by the content creator, and 2) that the sender can properly control the behavior of the receiver. The latter is essentially providing an open-loop flow control mechanism, a requirement for a specification that covers broadcast channels (without an upstream, or feedback, channel). Assuming a finite set of buffer resources at the receiver, by proper clock recovery and timestamping of events the source can always ensure that these resources are not exhausted.

Clock recovery is typically performed using clock references. The receiving system has a local system clock which is controlled by a PLL, driven by the differences in received clock references and the local clock references at the time of their arrival. This way the receiver's clock speed is increased or increased, matching that of the sender. In addition, coded units are associated with decoding time stamps, indicating the time instance in which a unit is removed from the receiving system's decoding buffer. The combination of clock references and time stamps is sufficient for full control of the receiver.

In order to properly address specification issues without making unnecessary assumption about system implementation, MPEG-4 Systems defines a System Decoder Model. This represents an idealized unit, in which operations can be unambiguously controlled and characterized. The MPEG-4 System Decoder Model exposes resources available at the receiving terminal, and defines how they can be controlled by the sender or content creator. The model is shown in Figure 26. It is composed of a set of decoders (for the various audio or visual object types), provided with two types of buffers: decoding and composition.

The decoding buffers have the same functionality as in previous MPEG specifications, and are controlled by clock references and decoding timestamps. In MPEG-2, each program had its own clock. Proper synchronization was ensured by using the same clock for coding and transmitting the audio and video components. In MPEG-4, each individual object is assumed to have its own clock, or Object Time Base (OTB). Of course, several objects may share the same clock. In addition, coded units of individual objects (Access Units – AUs, corresponding to an instance of a video object or a set of audio samples) are associated with Decoding Timestamps (DTSs). Note that the decoding operation at DTS is considered (in this ideal model) to be instantaneous.

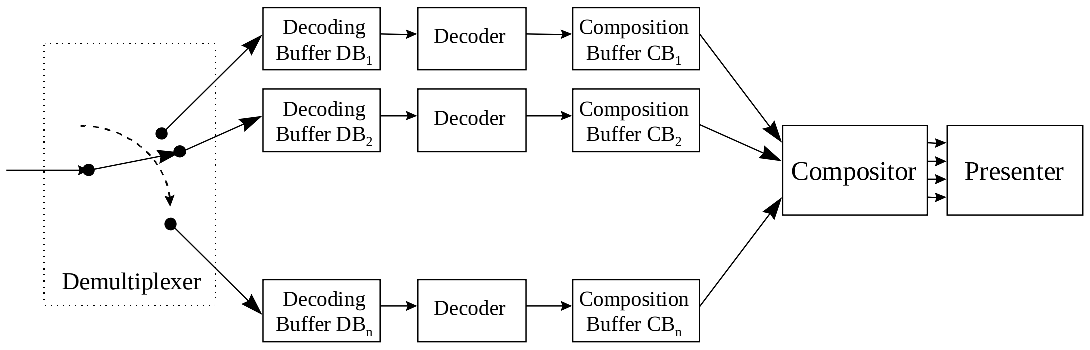


Figure 26 Systems Decoder Model

The composition buffers which are present at the decoder outputs form a second set of buffers. Their use is related to object persistence. In some situations, a content creator may want to reuse a particular object after it has been presented. By exposing a composition buffer, the content creator can control the lifetime of data in this buffer for later use. This lifetime is controlled by an Expiration Timestamp (ETS). A decoded object is assumed to remain in this buffer until its ETS is reached, and can be used repeatedly until that time. This feature may be particularly useful in low bandwidth wireless environments. MPEG-4 defines an additional timestamp, the Composition Timestamp (CTS), which defines the time at which data is taken from the composition buffer for (instantaneous) composition and presentation.

In order to coordinate the various objects, a single System Time Base is assumed to be present at the receiving system. All object time bases are subsequently mapped into the system time base, so that a single notion of time exists in the terminal. For clock recovery purposes, a single stream must be designated as the master. The current specification does not indicate the stream that has this role, but a plausible candidate is the one which contains the scene description. Note also that, in contrast with MPEG-2, the resolution of both the STB and the OCRs is not mandated by the specification. In fact, the size of the OCR fields for individual Access Units is fully configurable, as we discuss later on.

In other designs (e.g., IETF's RTP), the assumption of a globally known clock can be made (provided by other network services such as NTP). There is work underway to provide a unified methodology so that mapping of MPEG-4 timing architecture can be seamlessly performed in such an environment as well. This is facilitated by the flexible multiplexing methodology adopted in MPEG-4, and is discussed later.

6.2 Scene Description

Scene Description refers to the specification of the spatio-temporal positioning and behaviour of individual of individual objects. It is a totally new component in the MPEG specifications, and allows the easy creation of compelling audiovisual content. Scene description involves an architectural component, i.e., the proper way to conceptually organize audiovisual information, and a syntactic component which is the mapping of such architecture into a bitstream. Note that the scene description is transmitted in a separate stream from the individual media objects. This allows one to change the scene description without operating on any of the constituent objects themselves.

6.2.1 Architecture

The architecture of MPEG-4's scene description is based on VRML [28,29]. Scenes are described as hierarchies of nodes forming a tree. Leafs of the tree correspond to media objects (audio or visual, natural or synthetic), while intermediate nodes perform operations on their underlying nodes (grouping, transformation, etc.). Nodes also expose attributes through which their behaviour can be controlled. This hierarchical design is shown in Figure 27.

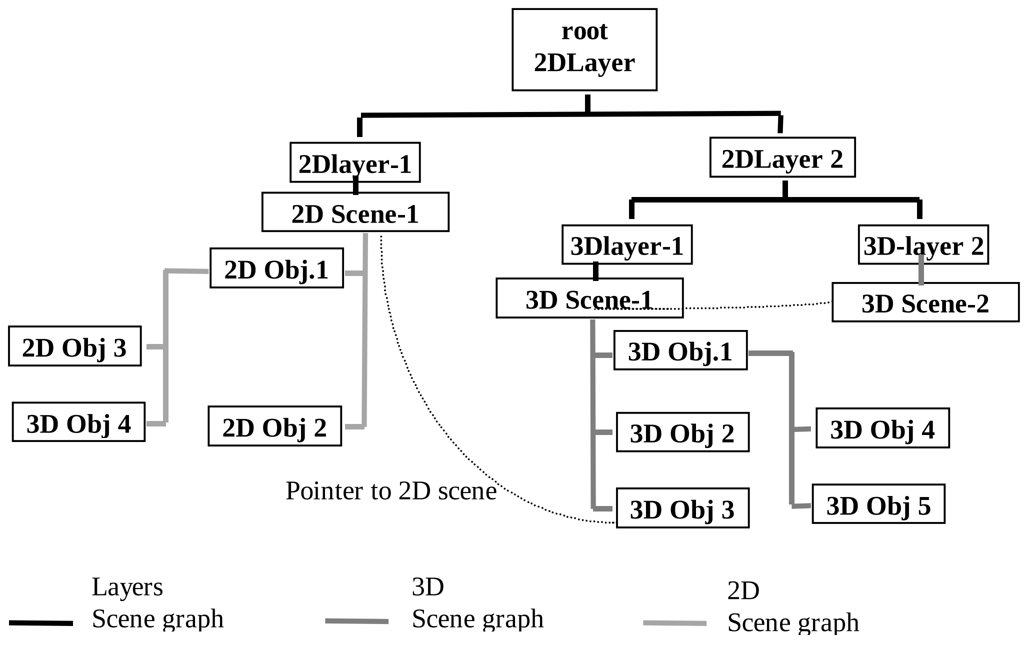


Figure 27 Scene Description

There are however several key differences between the domains that VRML and MPEG-4 address, which necessitate the adoption of slightly different approaches. MPEG-4 is concerned with the description of highly dynamic scenes, where temporal evolution is more dominant than navigation. VRML, on the other hand, allows the definition of a static 3-D world (static in the sense that all the objects in that world are predefined and cannot be changed) in which a user is allowed to navigate. As a result, scene descriptions in MPEG-4 actually have their own time base, and can be updated at any time. Updates can take the form of a node's replacement, elimination, insertion, or attribute value modification. The presence of a time base and decoding time stamps ensures that application of such updates at the correct time instances.

In addition, MPEG-4 also needs to address pure 2-D composition of objects. The complexity and cost of 3-D graphics versus the possibility of developing low-cost or low-power systems makes it desirable to partition the space of graphics capabilities. In Figure 19 we see an example where both 2-D and 3-D components coexist.

As a result of the close relationship of the MPEG-4 scene description with VRML, the types of scene description capabilities provided by MPEG-4 are essentially those provided by VRML nodes. It is currently examined if MPEG-4 can be cast as an extension of VRML, with appropriately defined subsets as MPEG-4 specific profiles.

6.2.2 Binary Format for Scenes

The mapping of the scene description into a parametric form, suitable for low-overhead transmission has resulted in a scene description format called Binary Format for Scenes (BIFS). This representation format associates each node with a node type. Nodes are then represented by their node type and a set of attribute value specifications. To avoid the specification of all attributes of a node, default values are used, and attributes are individually addressable within a node. This way one can, for example, only specify a non-default value for attribute X of a node of type Y.

Furthermore, nodes can optionally be reused. In this case, they are associated with an identifier which can appear in place of a node of the same type. In Figure 27, for example, the node "3D Obj 3" contains the child node "2D Scene-1" which is used elsewhere in the scene as well. As mentioned earlier, scene descriptions can be updated: nodes can be inserted, deleted, replaced, or their attributes can be modified.

In addition to the VRML set of nodes, MPEG-4 defines its own set of nodes, particularly to handle media objects including natural audio and video, still images, face animation, basic MIDI, text-to-speech synthesizer, streaming text, 2-D composition operators, layout control, etc.

6.2.3 Interactivity

Interactivity in MPEG-4 can take two forms, client-based and server-based. In the client-based case, user operations affect the local scene description. The VRML model of events and routes is used within the BIFS format, thus providing direct support for a large variety of circumstances. In addition, and noting that user events can be transformed to scene updates, one can also have a form of interactivity that does not necessitate normative support by the specification: user events can form a secondary source of scene updates.

Server-based interaction requires the presence of an upstream channel. The details of such a mode of interactivity are still under investigation, as they are slightly complicated by the needs for network-independence.

6.2.4 Adaptive Audio-Visual Session

An additional, flexible mechanism for describing scenes is also being developed, called Adaptive Audio-Visual Session (AAVS). This is based on the use of the Java language for constructing scenes, but not for composition, rendering, or decoding. By taking the programmable aspect of the system outside of the main data flow of decoders and the rendering engine (which have to operate extremely fast), overall performance can be kept high. Note that the AAVS approach is being designed as an extension of BIFS. In that sense, a BIFS scene can be a subset of an AAVS scene (but not vice versa).

The benefits of the AAVS approach are in three major categories. First, because of its flexible nature, the scene description is capable of adapting to terminal capabilities hence providing a form of graceful degradation. Secondly, when scene description operations can be expressed concisely in a flexible way (e.g., a spline trajectory), using a flexible approach may result in increased compression. Finally, by allowing programmability, user interaction can extend beyond the modes defined in a parametric scene description and become as rich as the content creator wishes.

Use of programmability in an audiovisual terminal certainly opens up a very broad spectrum of opportunities. If the integration with the requirements of the real-time world of decoders is successful, then this will be a major step forward in information representation.

6.3 Associating Scene Description with Elementary Streams

6.3.1 Object Descriptors

Individual object data as well as scene description information is carried in separate Elementary Streams (ES). As a result, BIFS media nodes need a mechanism to associate themselves with the ESs that carry their data (coded natural video object data etc.). A direct mechanism would necessitate the inclusion of transport-related information into the scene description. As we mentioned earlier, an important requirement in MPEG-4 is transport-independence. As a result, an indirect way was adopted, using Object Descriptors (ODs).

Each media node is associated with an object identifier, which in turn uniquely identifies an OD. Within an OD, there is information on how many ESs are associated with this particular object (may be more than one for scalable video/audio coding, or mutli-channel audio coding), and information describing each of those streams. The latter information includes the type of the stream, as well as how to locate it within the particular networking environment used. This approach simplifies remultiplexing (e.g., going through a wired-wireless interface), as there is only one entity that may need to be modified.

6.3.2 Stream Map Table

The object descriptor allows unique reference of an elementary stream by an id; this id may be assigned by an application layer when the content is created. The transport channel in which this stream is carried may only be assigned at a later time by a transport entity; it is identified by a channel association tag associated to an elementary stream id by a stream map table. In interactive applications, the receiving terminal may select the desired elementary streams, send a request and receive the stream map table in return. In broadcast and storage applications, the complete stream map table must be included in the applications signalling channel.

6.4 Multiplexing

The key underlying concept in the design of the MPEG-4 multiplexer is network independence. MPEG-4 content may be delivered across a wide variety of channels, from very low bit rate wireless, to high-speed ATM and broadcast systems, to DVDs. A critical design question was what should be the tools included in the specification for mandatory implementation. Clearly, the broad spectrum of channels could not allow a single solution to be used. At the same time, inclusion of a large number of different tools and configuration would make implementations extremely complex and – through excessive fragmentation through profiles – make interoperability extremely hard to achieve in practice. Consequently, the assumption was made that MPEG-4 would not provide specific transport-layer features but would instead make sure that it could be easily mapped to existing such layers. This is accomplished by allowing several components of the multiplexer to be configurable, thus allowing designers to achieve the desired trade-off between functionality and efficiency. In addition, it is assumed that Quality of Service (QoS) guarantees may be made available by the underlying transport service if so desired.

The overall multiplexing architecture of MPEG-4 is shown in Figure 28. At the top-most level, we have the Access Unit Layer (AL) which provides the basic conveyor of timing and framing information. It is at this level where timestamps and clock references are provided. The AL header, however, is very flexible: the presence of OCR/DTS is optional, and their resolution (number of bits) is configurable. In addition, the header contains information about framing (start or end of a coded unit, random access indicator), as well as sequence numbering. The latter is particularly useful in error-prone wireless or broadcast environments, where preemptive retransmission of critical information (e.g., scene description) may be performed. Using a sequence number a receiver that has already accurately received the information can ignore the duplicate. In order to be able to “bootstrap” the demultiplexing process, the AL header configuration information is carried in a channel that has a predefined configuration.

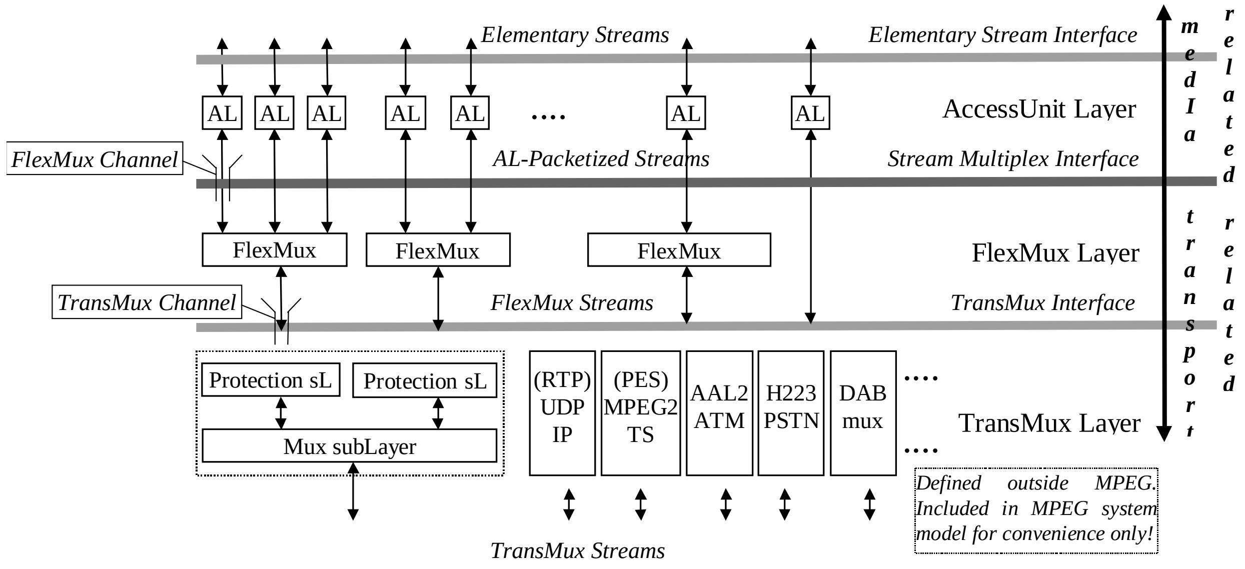
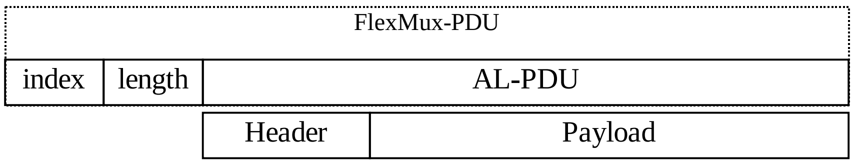
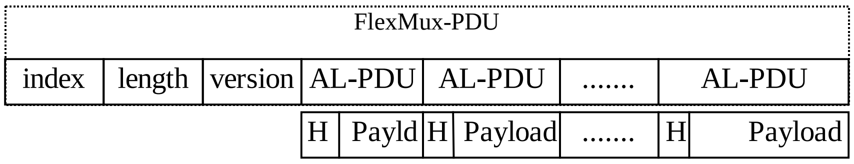


Figure 28 MPEG-4 Multiplexing Architecture

Immediately below the AL we have the optional Flexible Multiplexer or “FlexMux” layer. This is a very simple design, intended for systems that may not provide native multiplexing services. An example is the data channel available in GSM cellular telephones. Its use, however, it entirely optional, and does not affect the operation of the rest of the system. As shown in the right side of Figure 28 there can be a “null” connection directly from the Access Unit Layer to the lower layers. The FlexMux provides two modes of operation, a “simple” and a “muxcode” mode, as shown in Figure 29.



(a) Simple (single-object PDU)



(b) MuxCode (multi-object PDU)

Figure 29 FlexMux Modes

In the simple mode, data from a single object (or scene description) is present in the FlexMux payload (appropriately encapsulated as an AL PDU). In the MuxCode mode, data from multiple objects are placed in predefined positions within the FlexMux payload. The ‘index’ field of the FlexMux header indicates which of the modes is used, depending on its value.

Finally, at the lowest level, we have the Transport Multiplexing or “TransMux” layer which is not specified by MPEG-4. This can be any current or future transport layer facility, including MPEG-2 Transport Stream, RTP, H.223, etc. For a mobile system, this layer must provide its own protection sublayer to ensure the desired QoS characteristics.

6.5 File Format

For storage-based content deliver (e.g., CD-ROM or DVD), MPEG-4 also defines a file format. This is a simple layer that substitutes the TransMux layer and where random access information is provided in the form of directories. This allows an application or user to rapidly move back and forth in an MPEG-4 file, having immediate access to Access Units. In order to easily identify MPEG-4 files as such, the string ‘MPEG-4’ is used as a magic number (the first 6 bytes in the file), and the use of the ‘.mp4’ extension has been adopted.

The details of file format are shown in [25]. Multiplexed elementary streams are encapsulated in a Stored File Format Layer. An MPEG-4 file is expected to contain a Header followed by Segment data that in turn contains a series of zero or more FlexMux PDUs. The segments form a doubly linked list for convenient navigation within the file. Each segment header consists of a list of directory entries which provide pointers to access units of elementary streams that are contained within its segment data. Note that at the discretion of author, random access may be provided to just a subset of access units contained in the segment, thus, not all units need to have directory entries.

6.6 Syntactic Description

Syntactic description in past MPEG specifications was performed using ad-hoc techniques (pseudo-C). While this provided a concise way of representing simple bitstreams, it could not fully describe the sophisticated constructs required in source coding (e.g., Huffman codes) without explanatory text , tables, etc. Furthermore, the description was not amenable to software tool manipulation. Indeed, in the 50-year history of media representation and compression the lack of software tools is particularly striking. This has made the task of both codec and application developers much more difficult.

Use of source coding, with its bit-oriented nature, directly conflicts with the byte-oriented structure of modern microprocessors and makes the task of handling coded audiovisual information more difficult. A simple example is fast decoding of variable length codes; every programmer that wishes to use information using entropy coding must hand-code the tables so that optimized execution can be achieved. General-purpose programming languages such as C++ and Java do not provide native facilities for coping with such data. Even though other facilities already exist for representing syntax (e.g., ASN.1 – ISO International Standards 8824 and 8825), they cannot cope with the intricate complexities of source coding operations (variable length coding etc.).

MPEG-4 Systems has adopted an object-based syntactic description language for the definition of its bitstream syntax (Flavor – Formal Language for Audio-Visual Object Representation [24, 31-33]). It is designed as an extension of C++ and Java in which the type system is extended to incorporate bitstream representation semantics (hence forming a syntactic description language). This allows the description, in a single place, of both the in-memory representation of data as well as their bitstream-level (compressed) representation as well. Also, Flavor is a declarative language, and does not include methods or functions. By building on languages widely used in multimedia application development, one can facilitate integration with an application’s structure.

Figure 30 shows a simple example of a Flavor representation. Note the presence of bitstream representation information right after the type within the class declaration. The map declaration is the mechanism used in Flavor to introduce constant or variable length code tables (1-to-n mappings); in this case, binary codewords (denoted using the ‘0b’ construct) are mapped to values of type unsigned char. Flavor also has a full complement of object-oriented features pertaining to bitstream representation (e.g., “bitstream polymorphism”) as well as flow control instructions (if, for, do-while, etc.). The latter are placed within the declaration part of a class, as they control the serialization of the class’ variables into a bitstream.

```
map SampleVLC(unsigned char) {
        0b0,  2,
        0b10, 5,
        0b11, 7
}

class HelloBits {
        int(8) size;
        int(size) value1;
        unsigned char(SampleVLC) value2;
}
```

Figure 30 A simple example of syntactic description.

A translator has been developed that automatically generates standard C++ and Java code from the Flavor source code [24], so that direct access to, and generation of, compressed information by application developers can be achieved with essentially zero programming. This way, a significant part of the work in developing a multimedia application (including encoders, decoders, content creation and editing suites, indexing and search engines) is eliminated.

7. PROFILES AND LEVELS

The concept of Profiles and levels of MPEG-2 Video [1,4] is extended to MPEG-4 such that profiles and levels provide a means of defining subsets of the syntax and semantics of MPEG-4 Video, Audio and Systems specifications and thereby the decoder capabilities required to decode a particular bitstream. A profile is a defined sub-set of the entire bitstream syntax that is defined by this Specification. A level is a defined set of constraints imposed on parameters in the bitstream. Conformance tests will be carried out against defined profiles at defined levels. The purpose of defining conformance points in the form of profiles and levels is to facilitate bitstream interchange among different applications. Implementers of MPEG-4 are encouraged to produce decoders and bitstreams which correspond to those defined conformance regions. The discretely defined profiles and levels are the means of bitstream interchange between different applications of MPEG-4.

In MPEG-4, it is likely that profiles will be defined for video objects, audio objects and systems. In addition it is likely that profiles may also be defined for audio composition and video composition. The work on profiles is ongoing and is thus likely to evolve; we now present the current structure of profiles envisaged and their key requirements [12,41,42]. Currently, three profiles each are being considered for each of the three parts, video, audio and system objects; the requirements for these profiles are being established.

The profiles being considered for video are as follows.

- Video Simple Object Profile
- Video Random Access Object Profile
- Video Main Object Profile

Similarly, the following three three profiles are being considered for audio.

- Audio Speech Object Profile
- Audio Low Delay Object Profile
- Audio Main Object Profile

The three profiles being considered for systems are as follows.

- Systems Simple Profile
- Systems Interactive Profile
- Systems Main Profile

8. VERIFICATION TESTS

MPEG-4 is planning to hold verification tests to confirm the performance of its various combination of tools that will form profiles. The first of the series of such tests [45] will take place in March 1998 and is aimed to verify error resilience tools.

Version 1 of the MPEG-4 video codec will be implemented in software before the October 1997 MPEG meeting. It will be combined with simulations of the Universal Access Layer, a component of the MPEG-4 System Layer, and a TransMux. The particular TransMux to be implemented depends upon the application and corresponding network. For wireless applications the TransMux will be ITU's mobile multiplexing standard, H.223/Mobile. The overall encoding system to be simulated for the demonstration of MPEG-4 over a wireless network consists of an application layer consisting of an MPEG-4 error resilient video encoder and simulated audio data, an access unit layer consisting of adaptation layers for audio and video data, H.223/Mobile layer consisting of adaptation layers and a multiplexer. The multiplexed data is to be stored on PC hard drive and passes through physical layer simulation to the decoding system that performs the inverse operations such as H.223/Mobile demultiplexing and adaptation layers for audio and video, access unit layer consisting of adaptation layer for audio and video data, and back to application layer which consists of simulated audio data and MPEG-4 error resilient video decoder. The test conditions being considered are shown in Table 7.

Table 7 Wireless Network

	PCS	**IMT2000 (1)**	**IMT2000 (2)**
Data rate (Total)	32 kbit/s	128 kbit/s	384 kbit/s
Error Conditions	Random and Burst	Random and Burst	Random and Burst

9. BEYOND CURRENT MPEG-4 WORK

As mentioned earlier, the MPEG-4 effort as described in the paper thus far has recently been labeled to be MPEG-4 Version 1. Within MPEG-4 a number of other tools are currently being investigated and expect to be mature a little later than expected, these tools are expected to be released in a following release and together with MPEG-4 Version 1, will be called MPEG-4 Version 2. MPEG-4 Version 2 is expected to include either tools that provide new functionalities not supported in MPEG-4 Version 1, or tools that even provide the same functionalities but a lot more efficiently, with much higher quality, or with lower implementation cost. Also, Version 2 is expected to be maintain backward compatibility with Version 1,

Besides MPEG-4, MPEG committee has recently accepted a new work item called "Multimedia content description interface," and is likely to be the basis of MPEG-7, the next MPEG standard. MPEG-7 has the primary goal to extend the limited capabilities of proprietary solutions for identifying content. MPEG-7 will specify [46] a standard set of descriptors that can be used to describe various types of multimedia information. This description shall be associated with the content itself, to allow fast and efficient searching for material of a user's interest. AV material that has MPEG-7 data associated with it, can be indexed and searched for.

10. SUMMARY

In this paper we have introduced the MPEG-4 standard currently in progress. The necessary background information leading up to the MPEG-4 standard and the multiple facets of this standard were discussed including the directions for future. In particular we have presented the following.

- Brief review of the low bit-rate ITU-T H.263 standard which is the starting basis for the MPEG-4 standard.
- Basics of MPEG-4 including discussion of its focus, applications, functionalities and requirements.
- The MPEG4 Video development process including test conditions, first evaluation, results of evaluation, formulation of core experiments and definition of the Verification Model.
- A brief introduction to the MPEG-4 Audio, SNHC and Systems development process.
- Details of coding tools and methods that are expected to form the MPEG-4 Visual standard.
- Introduction to coding methods and tools that are expected to form the MPEG-4 Audio standard.
- Discussion of tools and techniques employed by MPEG-4 Systems standard.
- The ongoing work on MPEG-4 profiles.
- A brief introduction to MPEG-4 Version 2, and the scope of MPEG-7, the next MPEG standard.

The success of MPEG-4 will eventually depend on many factors such as the market needs, competing standards, software versus hardware paradigms, complexity versus functionality tradeoffs, timing, profiles etc. Technically, MPEG-4 appears to have a significant potential due to the integration of natural and synthetic worlds, computers and communication applications, and the functionalities and flexibilities it offers. Initially, perhaps only the very basic functionalities will be useful. As the demand for sophisticated multimedia grows the advanced functionalities may be useful. Up-to-date information on progress on various topics in MPEG-4 can be found by visiting MPEG related websites [48-52].

REFERENCES

[1] B. G. Haskell, A. Puri, and A. N. Netravali, “Digital Video: An Introduction to MPEG-2,” Chapman & Hall, ISBN 0-412-08411-2, 1997.

[2] ITU-T Recommendation H.261, Video Codec for Audio Visual Services at p×64 kbit/s, 1990.

[3] ISO/IEC 11172 International Standard (MPEG-1), Information Technology – Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to About 1,5 Mbit/s, 1993.

[4] ISO/IEC 13818 International Standard (MPEG-2), Information Technology – Generic Coding of Moving Pictures and Associated Audio (also ITU-T Rec. H.262), 1995.

[5] ITU-T, “Draft ITU-T Recommendation H.263: Video Coding for Low Bitrate Communication,” Dec. 1995.

[6] AOE Group, “MPEG-4 Proposal Package Description (PPD) - Rev. 3,” ISO/IEC JTC1/SC29/WG11 N0998, Tokyo, July 1995.

[7] AOE Group, “MPEG-4 Testing and Evaluation Procedures Document,” ISO/IEC JTC1/SC29/WG11 N0999, Tokyo, July 1995.

[8] L. Chiariglione, “MPEG-4 Call for Proposals,” ISO/IEC JTC1/SC29/WG11 N0997, Tokyo, July 1995.

[9] A. Puri, “Status and Direction of the MPEG-4 Standard,” International Symposium on Multimedia and Video Coding, New York, October 1995; also published in a book by Plenum Press.

[10] T. Ebrahimi, “Report of Ad hoc Group on Definition of VMs for Content Based Video Representation,” ISO/IEC JTC1/SC29/WG11 MPEG 96/0642, Munich, Jan. 1996.

[11] A. Puri, “Report of Ad hoc Group on Coordination of Future Core Experiments in MPEG-4 Video,” ISO/IEC JTC1/SC29/WG11 MPEG 96/0669, Munich, Jan. 1996.

[12] MPEG-4 Requirements Ad hoc Group, “Draft of MPEG-4 Requirements,” ISO/IEC JTC1/SC29/WG11 N1238, Florence, March 1996.

[13] MPEG-4 Video Group, “MPEG-4 Video Verification Model Version 2.0,” ISO/IEC JTC1/SC29/WG11 N1260, Florence, March 1996.

[14] MPEG-4 Integration Group,”MPEG-4 SNHC Call For Proposals,” ISO/IEC JTC1/SC29/WG11 N1195, Florence, March 1996.

[15] MPEG-4 Integration Group, “MPEG-4 SNHC Proposal Package Description,” ISO/IEC JTC1/SC29/WG11 N1199, Florence, March 1996.

[16] H. Peterson, “Report of the Ad Hoc group on MPEG-4 Video Testing Logistics,” ISO/IEC JTC1/SC29/WG11 Doc. MPEG95/0532, Nov. 1995.

[17] A. Puri, “MPEG-4: A Flexible and Extensible Multimedia Coding Standard in Progress,” invited paper in IEEE Multimedia book, 1996.

[18] A. Puri, A. R. Reibman, R. L. Schmidt and B. G. Haskell, “Robustness Considerations in ISO MPEG-4 and ITU-T Mobile Video Standards,” Proceedings MoMuC-3, Princeton, Sept 1996; also revised and published in a book by Plenum Press, 1997.

[19] D. Lindbergh, “The H.324 Multimedia Communication Standard,” IEEE Communications Magazine, vol. 34, no. 12, pp. 46-51, Dec. 1996.

[20] ITU-T, “ITU-T Recommendation H.223: Multiplexing Protocol for Low Bitrate Multimedia Communication,” 1995.

[21] ITU-T, “ITU-T Recommendation H.223 Annex A: Multiplexing Protocol for Low Bitrate Mobile Multimedia Communication,” 1996.

[22] ITU-T H.263+ Video Group, “Draft 12 of ITU-T Recommendation H.263+,” May, 1997.

[23] B. G. Haskell, A. Puri and J. Osterman, “Happenings in ISO MPEG: An Introduction to MPEG-4,” invited presentation at Data Compression Conference, Snow Bird, March 1997.

[24] 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.

[25] MPEG-4 Systems Group,“MPEG-4 Systems Working Draft Version 5.0,” ISO/IEC JTC1/SC29/WG11 N1825, Stockholm, July 1997.

[26] MPEG-4 Video and SNHC Groups, “MPEG-4 Visual Working Draft Version 4.0,” ISO/IEC JTC1/SC29/WG11 N1797, Stockholm, July 1997.

[27] MPEG-4 Audio Group, “MPEG-4 Audio Working Draft Version 4.0,” ISO/IEC JTC1/SC29/WG11 N1745, Stockholm, July 1997.

[28] “ISO/IEC 14472 Draft International Standard: Virtual Reality Modeling Language,” 1997.

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

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

[31] Y. Fang and A. Eleftheriadis, “A Syntactic Framework for Bitstream-Level Representation of Audio-Visual Objects,” *Proceedings, 3rd IEEE International Conference on Image Processing* (ICIP-96), Lausanne, Switzerland, September 1996.

[32] A. Eleftheriadis, “The MPEG-4 System Description Language: From Practice to Theory,” *Proceedings, 1997 IEEE International Conference on Circuits and Systems* (ISCAS-97), Hong Kong, June 1997.

[33] A. Eleftheriadis, “Flavor: A Language for Media Representation,” *Proceedings, ACM Multimedia ’97 Conference*, November 1997 (to appear).

[34] Signal Processing: Image Communication, Special Issue on MPEG-4, Part 1: Invited Papers, Vol. 10, Nos. 1-3, May 1997.

[35] Signal Processing: Image Communication, Special Issue on MPEG-4, Part 2: Submitted Papers, Vol. 10, No. 4, July 1997.

[36] IEEE Trans. on Circuits and Systems for Video Technology, Special Issue on MPEG-4, Vol. 7, No. 1, February 1997.

[37] MPEG-4 Video Group, “MPEG-4 Video Verification Model Version 7.0,” ISO/IEC JTC1/SC29/WG11 N1642, Bristol, April 1997.

[38] A. Puri, R. L. Schmidt and B. G. Haskell, “Improvements in DCT Based Video Coding,” Proc. SPIE Visual Communications and Image Processing, San Jose, Feb. 1997.

[39] P. J. L van Beek, A. M. Tekalp and A. Puri, “2-D Mesh Geometry and Motion Compression for Efficient Object Based Video Compression,” to appear in IEEE Int. Conf. On Image Processing, Oct. 1997.

[40] L. Chiariglione, “Resolutions of 40th WG11 meeting,” ISO/IEC JTC1/SC29/WG11 N1716, Stockholm, July 1997.

[41] Requirements Group, “MPEG-4 Requirements version 4,” ISO/IEC JTC1/SC29/WG11 N1727, Stockholm, July 1997.

[42] Requirements Group, “MPEG-4 Profile Requirements version 4,” ISO/IEC JTC1/SC29/WG11 N1728, Stockholm, July 1997.

[43] Requirements Group, “MPEG-4 Applications Document,” ISO/IEC JTC1/SC29/WG11 N1729, Stockholm, July 1997.

[44] Requirements Group, “MPEG-4 Overview,” ISO/IEC JTC1/SC29/WG11 N1730, July 1997.

[45] Ad hoc Group on Error Resilience Core Experiments, “Plan for March 1998 Error Resilience Verification Test,” ISO/IEC JTC1/SC29/WG11 N1829, Stockholm, July 1997.

[46] Requirements Group, “MPEG-7 Context and Objectives version4,” ISO/IEC JTC1/SC29/WG11 N1733, Stockholm, July 1997.

[47] R. Cox, B. Haskell, Y. LeCun, B. Shahraray and L. Rabiner, “On the Applications of Multimedia Processing to Communications,” AT&T internal technical memo; to appear in IEEE Transactions.

[48] ISO/IEC JTC1/SC29/WG11 (MPEG) Web Site: http://drogo.cselt.it/mpeg.

[49] MPEG-4 Systems Web Site: http://garuda.imag.fr/MPEG4.

[50] Flavor Web Site: http://www.ee.columbia.edu/flavor.

[51] MPEG-4 Video Web Site: http://wwwam.hhi.de/mpeg-video

[52] MPEG-4 SNHC Web Site: http://www.es.com/mpeg4-snhc.