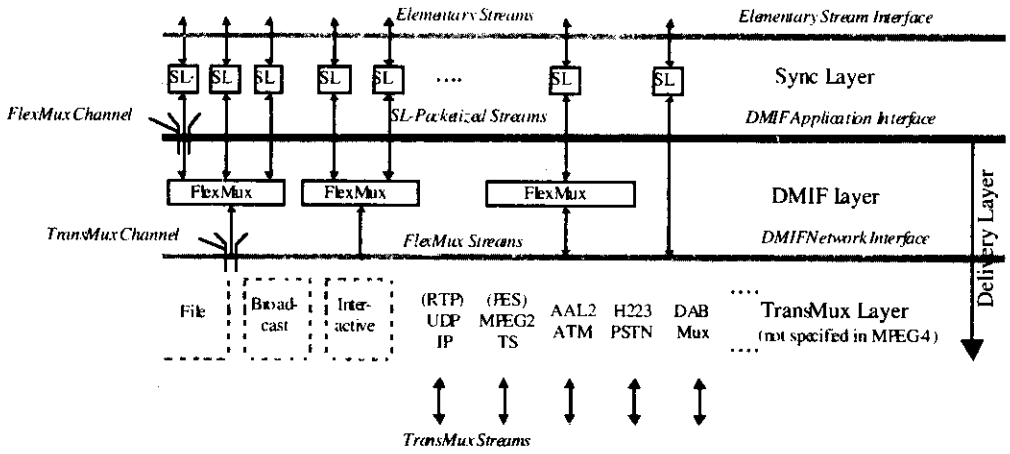

Figure 5.26 The MPEG-4 Systems layer model [5.36]. ©2001 ISO/IEC.

elementary streams and recovery of the media objects or scene-description time base. It also enables synchronization among them. The synchronized delivery of streaming information from source to destination, exploiting different QoS as available from the network, is specified in terms of the synchronization layer and a delivery layer containing a two-layer multiplexer as depicted in Figure 5.26. The first multiplexing layer is managed according to the DMIF. This multiplex may be embodied by the MPEG-defined FlexMux tool, which allows grouping of Elementary Streams (ESs) with a low multiplexing overhead. Multiplexing at this layer may be used. An example is grouping ES with similar QoS requirements, reducing the number of network connections or the end-to-end delay. The transport multiplexing (TransMux) layer models the layer that offers transport services matching the requested QoS. Only the interface to this layer is specified by MPEG-4, and the concrete mapping of the data packets and control signaling must be done in collaboration with the bodies that have jurisdiction over the respective transport protocol. Use of the FlexMux multiplexing tool is optional, and this layer may be empty of the underlying TransMux instance providing all the required functionality. However, the synchronization layer is always present. With regard to the MPEG-4 Systems layer model it is possible to do the following [5.38]:

- Identify access units, transport timestamps and clock reference information and identify data loss
- Optionally interleave data from different ESs into FlexMux streams
- Convey control information
- Indicate the required QoS for each ES and FlexMux stream
- Translate such QoS requirements into actual network resources
- Associate ESs to media objects
- Convey the mapping of elementary streams to FlexMux and TransMux channels

In general, the user observes a scene that is composed following the design of the scene's author. Depending on the degree of freedom allowed by the author, the user has the possibility to interact with the scene. It is also important to have the possibility to identify the intellectual property in MPEG-4 media objects. To support this, MPEG has worked with representatives of different creative industries in the definition of syntax and tools. MPEG-4 incorporates identification of the intellectual property by storing unique identifiers that are issued by international numbering systems. These numbers can be applied to identify a current rights holder of a media object.

MPEG-4 Version 1

The MPEG-4 requirements have been addressed by the six parts of the finalized MPEG-4 Version 1 standard.

Part 1: Systems—Specifies scene description, multiplexing, timing identification, synchronization and recovery mechanisms, buffer management and management and protection of intellectual property [5.39].

Part 2: Visual—Specifies the coded representation of natural and synthetic visual objects (computer-generated scenes). This will, for example, allow the virtual presence of videoconferencing participants. The tools and algorithms of the MPEG-4 Visual standard will support bit rates (typically between 5 Kb/s and 10 Mb/s), formats (progressive as well as interlaced video) and resolutions (typically from sub-QCIF to beyond TV). Also, compression efficiency, content-based functionalities; scalability of textures, images and video; shape and alpha channel coding; robustness in error-prone environments and face animation will be supported [5.40].

Part 3: Audio—Specifies the coded representation of natural and synthetic audio objects. It facilitates a wide variety of applications that could range from intelligible speech to high-quality multichannel audio and from natural sounds to synthesized sounds. In particular, it supports the highly efficient representation of audio objects, consisting of speech signals, synthesized speech, general audio signals and synthesized audio and bounded-complexity synthetic audio [5.35].

Part 4: Conformance testing—Defines conformance conditions for bit streams and devices. This part is used to test MPEG-4 implementations [5.41].

Part 5: Reference software—Includes software corresponding to most parts of MPEG-4 (normative and non-normative). It can be used for implementing compliant products [5.42].

Part 6: DMIF—Defines session protocol for the management of multimedia streaming over generic delivery technologies [5.43].

Parts 1 through 3 and 6 specify the core MPEG-4 technology, and Parts 4 and 5 are supporting parts. Parts 1, 2 and 3 are delivery independent and leave to Part 6 (DMIF) the task of dealing with the delivery layer. Although the various MPEG-4 parts are rather independent and thus can be used by themselves, they were developed to achieve maximum benefit results when they are used together. In October 1998, the first set of MPEG-4 standards was frozen. Work on MPEG-4 continued for Version 2, which adds tools to the MPEG-4 standard in the form of new profiles.

MPEG-4 Version 2

Version 2 is a backward compatible extension of Version 1. Figure 5.27 depicts the relationship between the two versions. MPEG-4 Version 2 is formally seen as amendments to the various parts of Version 1. The Systems layer of Version 2 is backward compatible with Version 1. In the areas of audio and visual, Version 2 adds profiles to Version 1 [5.35].

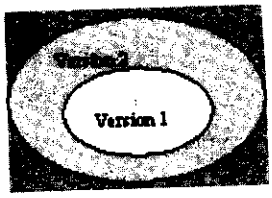

Figure 5.27 Relationship between MPEG-4 Versions 1 and 2.

Version 2 of the MPEG-4 systems extends Version 1 to cover issues like extended Binary Format for Scene Description (BIFS) functionalities and Java programming language (MPEG-J) support. Version 2 also specifies a file format to store MPEG-4 content. The version 2 BIFS (advanced BIFS) includes the following functionalities: advanced environment modeling in interactive virtual scenes; body animation of either a default body model present at the decoder or of a downloadable body model; chroma keying, which is used to generate a shape mask and a transparency value for an image or a video sequence; inclusion of hierarchical 3D meshes to BIFS scenes and associating interactive commands to media nodes. As opposed to the parametric system offered by MPEG-4 Version 1, MPEG-J is a programmatic system that specifies an Application Programming Interface (API) for interoperation of MPEG-4 media players with Java programming language code. Location of interfaces in the architecture of an MPEG-J enabled MPEG-4 system is shown in Figure 5.28. The lower half of this drawing depicts the parametric MPEG-4 system player, also referred to as the presentation engine. The MPEG-J subsystem controlling the representation engine, also referred to as the application engine, is depicted in the upper half of the drawing. The Java programming language application is delivered as a separate elementary stream to the MPEG-4 terminal. For a scene-graph API, the objective is to provide access to the scene graph: to inspect the graph, to alter nodes and their field and to add and remove nodes within the graph. The resource manager API is used for regulation of performance. It provides a centralized facility for managing resources. The terminal capability API is used when program execution is contingent upon the terminal configuration and its capabilities, both static (they do not change during execution) and dynamic. The media decoder API allows the control of the decoders that are present in the terminal. The network API provides a way to interact with the network and is compliant to the MPEG-4 DMIF application interface. Complex applications and enhanced interactivity are possible with these basic packages.

MPEG-4 Visual Version 2 adds technology in the following areas:

- Increased flexibility in object-based scalable coding
- Improved coding efficiency

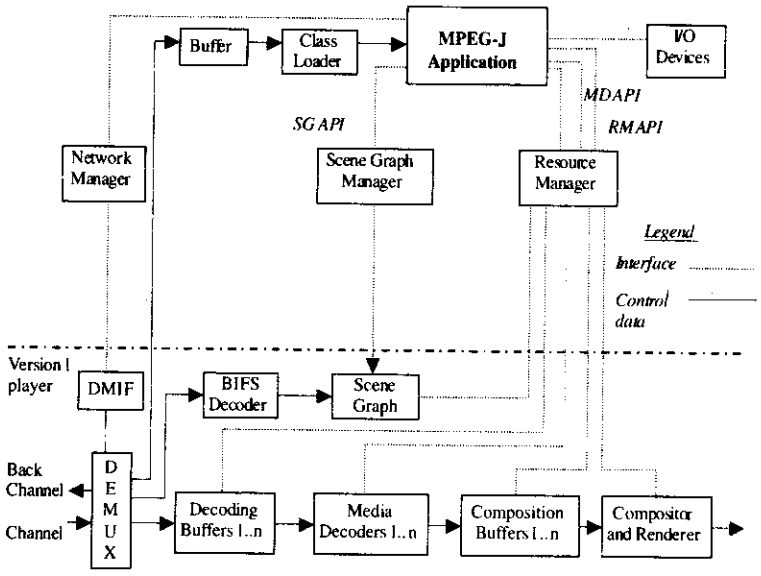

Figure 5.28 Location of interfaces in the architecture of an MPEG-J enabled MPEG-4 system [5.36]. ©2001 ISO/IEC.

- Improved temporal resolution stability with the low buffering delay
- Improved error robustness
- Added coding of multiple views. Intermediate views or stereoscopic views are supported based on the efficient coding of multiple images or video sequences. A particular example is the coding of stereoscopic images or video by redundancy reduction of information contained between the images of different views.

This version improves the motion estimation and compensation of rectangular and arbitrary-shaped objects and significantly improves the texture coding of arbitrary-shaped objects. In the area of motion estimation and compensation, two new technologies are introduced: Global Motion Compensation (GMC) and quarter-pel motion compensation. By GMC, we mean a method based on global motion estimation, image warping, motion trajectory coding and texture coding for prediction errors. On the other hand, quarter-pel motion compensation enhances the precision of the motion compensation scheme, at the cost of only small syntactical and computational overhead. An accurate motion description leads to a smaller prediction error and, hence, to better visual quality. In the area of texture coding, the Shape-Adaptive DCT (SA-DCT) improves the coding efficiency of arbitrary-shaped objects. The SA-DCT algorithm is based on predefined orthonormal sets of 1D DCT basis functions. Subjective evaluation tests within MPEG have shown that the combination of these techniques can result in a bit stream saving of up to 50% compared with Version 1, depending on the content type and data rate.

Another new technique is Dynamic Resolution Conversion (DRC). This is a way to stabilize the transmission buffering delay by minimizing the jitter of the amount of the coded output bits per Video Object Plane (VOP). Large frame skips are also prevented and the encoder can control the temporal resolution even in highly active scenes. The following three new tools for coding textures and still images are provided in Version 2:

- Wavelet tiling allows an image to be divided into several tiles and each tile to be encoded independently. This means that large images are encoded or decoded with very low memory requirements, and that random access at the decoder is significantly enhanced.
- Scalable shape coding allows encoding of arbitrary-shaped textures and still images in a scalable fashion. Using this tool, a decoder can decode an arbitrary-shaped image at any desired resolution. This tool enables applications to employ object-based, spatial and quality scalabilities at the same time.
- The error resilience tool adds new error-resiliency features. Using packetization and segment marker techniques, it significantly improves the error robustness in applications, such as image transmission across mobile channels and the Internet.

In Version 2, the use of multiple alpha channels to transmit auxiliary components is introduced. The basic idea is that the gray-scale shape is not only used to describe the transparency of the video object, but can be defined in a more general way. A gray-scale shape may represent the following:

- Transparency shape
- Disparity shape for multiview video objects (horizontal and vertical)
- Depth shape (acquired by laser range finder or by disparity analysis)
- Infrared or other secondary texture

All the alpha channels can be encoded by the shape-coding tools, that is, the binary shape-coding tool and the gray-scale shape-coding tool that employs a motion-compensated DCT and that usually has the same shape and resolution as the texture of the video object.

The body is an object capable of providing virtual body models and animations in the form of a set of 3D polygonal meshes ready for rendering. Two sets of parameters are defined for the body: Body Definition Parameter (BDP) set and Body Animation Parameter (BAP) set. The BDP set defines the set of parameters to transform the default body to a customized body with its body surface, body dimensions and, optionally, texture. If correctly interpreted, the BAPs will produce reasonably similar high-level results in terms of body posture and animation on different body models, without the need to initialize or calibrate the model. The human body model should be capable of supporting various applications, from realistic simulation of human motions to network games using simple human-like models. The work on body animation includes the assessment of the emerging standard as applied to hand signs for the hearing impaired.

Version 2 MPEG-4 provides a suite of tools for coding 3D polygonal meshes. Polygonal meshes are widely used as a generic representation of 3D objects. Capabilities for 3D mesh coding include the following:

- Coding of generic 3D polygonal meshes enables the efficient encoding of 3D polygonal meshes. The coded representation is generic enough to support both manifold and nonmanifold meshes.
- Incremental representation enables a decoder to reconstruct a number of faces in a mesh proportional to the number of bits in the bit stream that have been processed.
- Error resilience enables a decoder to recover a mesh partially when subsets of the bit stream are missing or corrupted.
- Level of Detail (LOD) scalability enables a decoder to reconstruct a simplified version of the original mesh containing a reduced number of vertexes from a subset of the bit stream. Such simplified representations are useful to reduce the rendering time of objects that are distant from the viewer, but they also enable less powerful rendering engines to render the object at a reduced quality.

MPEG-4 Audio Version 2 is an extension to MPEG-4 Audio Version 1. It adds new tools and functionalities to the MPEG-4 standard, but none of the existing tools of Version 1 is replaced. The following additional functionalities are provided by MPEG-4 Audio Version 2: error robustness, low-delay audio coding, parametric audio coding, CELP, silence compression, environmental spatialization and back channel audio transparent stream. The error-robustness tools provide improved performance on error-prone transmission channels. They can be distinguished into coded specific resilience tools and a common error-protection tool. The error-resilience tools reduce the perceived deterioration of the decoded audio signal that is caused by corrupted bits in the bit stream. The error resilient bit stream payload syntax is mandatory for all Version 2 object types. The Error Protection (EP) tool provides error protection applicable for a wide range of channel error conditions. The main features of the EP tool are as follows:

- Providing a set of error correcting and detecting codes with wide and small-step scalabilities in performance and in redundancy
- Providing a generic and bandwidth-efficient error protection framework that covers both fixed-length frame bit streams and variable-length frame bit streams
- Providing an Unequal Error Protection (UEP) configuration control with low overhead

MPEG-4 Audio Version 2 coding algorithms provide a classification of each bit stream field according to its error sensitivity. Based on this, the bit stream is divided into several classes, which can be separately protected by the EP tool so that more error-sensitive parts are strongly protected.

While the MPEG-4 general audio coder provides very efficient coding of general audio signals at low bit rates, it has an algorithmic encoding or decoding delay of up to several 100/ms and is thus not well suited for applications requiring low coding delay, such as real-time bidirectional communication. As an example, for a general audio coder operating at a 24 KHz sampling rate and 24 Kb/s, this results in an algorithmic coding delay of about 110 ms plus up to additional 210 ms for the use of the bit reservoir. To enable coding of general audio signals with an algorithmic delay not exceeding 20 ms, MPEG-4 Version 2 specifies a low-delay audio coder, which is derived from MPEG-2 and MPEG-4 AAC. Compared to speech-coding schemes, this coder allows compression of general audio signal types, including music, at a low delay. It operates at up to a 48 KHz sampling rate and uses a frame length of 512 or 480 samples, compared to 1,024 or 960 samples used in standard MPEG-2 or 4 AAC. Also, the size of the window used in the analysis and in the synthesis filter banks is reduced by a factor of two.

The parametric audio coding tools combine very low bit-rate coding of general audio signals with the possibility of modifying the playback speed or pitch during decoding without the need for an effects-processing unit. In combination with the speech and audio coding tools of Version 1, improved overall coding efficiency is expected for applications of object-based coding, allowing selection and/or switching between different coding techniques. Parametric audio coding uses the Harmonic and Individual Lines plus Noise (HILN) technique to code general audio signals at bit rates of 4 Kb/s and higher using a parametric representation of the audio signal. The basic idea of this technique is to decompose the input signal into audio objects, which are described by appropriate source models and represented by model parameters. Object models for sinusoids, harmonic tones, and noise are used in the HILN coder. Verification tests have shown that HILN coding has performance comparable to the other MPEG-4 coding technology operating at similar bit rates while providing the additional capability of an independent audio signal speed or pitch change when decoding. The tests have also shown that the scalable HILN coder provides quality comparable to that of a fixed-rate HILN coder at the same bit rate.

The silence compression reduces the average bit rate thanks to a lower bit-rate compression for silence. In the encoder, a voice activity detector is used to distinguish between regions with normal speech activity and those with silence or background noise. During normal speech activity, the CELP coding in Version 1 is used.

The environmental spatialization tools enable composition of an audio scene with a more natural sound source and sound environment modeling than is possible in Version 1. Both physical and perceptual approaches to spatialization are supported. The physical approach is based on a description of the acoustical properties of the environment (for example, room geometry, material properties and position of sound source) and can be used in applications like 3D virtual reality. The perceptual approach, on the other hand, permits a high-level perceptual description of the audio scene based on parameters similar to those of reverberations like movies. Although the environmental spatialization tools are related to audio, they are part of the BIFS in MPEG-4 System and are referred to as Advanced Audio BIFS. The back channel allows a request of client and/or client terminal to server. With this capability, interactivity can be achieved. In MPEG-4

System, the need for an upstream channel (back channel) is signaled to the client terminal by supplying an appropriate elementary stream descriptor declaring the parameters for that stream. In MPEG-4 Audio, the back channel allows feedback for bit-rate adjustment, scalability and error-protection adaptation.

The MPEG-4 Audio transport stream defines a mechanism to transport MPEG-4 Audio streams without using MPEG-4 Systems and is dedicated to audio-only applications. The transport mechanism uses a two-layer approach, namely a multiplex layer and a synchronization layer. The multiplex layer manages multiplexing of several MPEG-4 Audio payloads and audio-specific configuration information. The synchronization layer specifies a self-synchronized syntax of the MPEG-4 Audio transport stream, which is called Low Overhead Audio Stream (LOAS). The interface format to a transmission layer depends on the conditions of the underlying transmission layer.

Extensions to MPEG-4 Beyond Version 2

MPEG is currently working on a number of extensions to Version 2. In the visual area, the following technologies are in the process of being added: fine-grain scalability, tools for use of MPEG-4 in the studio, 2D and 3D animation coding, and digital cinema. Fine-grain scalability is a tool that allows small quality steps by adding or deleting layers of extra information. It is useful in a number of environments, notably for streaming purposes, but also for dynamic (statistical) multiplexing of pre-encoded content in broadcast environments. For the tools in the studio, care has been taken to preserve some form of compatibility with MPEG-2 profiles. Features such as 2D and 3D animation coding are under study. Digital cinema applications will require truly lossless coding [5.35].

Advanced BIFS provides new nodes to be used in the scene graph for monitoring available media and for managing media, such as sending commands to a server, providing advanced control of media playback, and using the so-called EXTERNPROTO, a node that provides further compatibility with Virtual Reality Modeling Language (VRML), which allows writing macros that define the behavior of objects. Also, it covers advanced compression of BIFS data, and, in particular, optimal compression for mesh and for arrays of data.

The extensible MPEG-4 Textual Format (XMT) is a framework for representing an MPEG-4 scene description using a textual syntax. XMT allows the content authors to exchange the content with other authors, tools or service providers and facilitates interoperability with both the Extensible 3D (X3D) being developed by the Web3D Consortium and the Synchronized Multimedia Integration Language (SMIL) from the Web3C Consortium.

The advanced synchronization model (usually called FlexTime) supports synchronization of objects from multiple sources with possibly different time bases. The FlexTime model specifies timing using a flexible, constraint-based timing model. In this model, media objects can be linked to one another in a time graph using relationship constraints such as CoStart, CoEnd, or Meet. In addition, to allow some flexibility to meet these constraints, each object may have a flexible duration with specific stretch and shrink mode preferences that may be applied.

Profiles in MPEG-4

Profiles exist for various types of media content (audio, visual and graphics) and for scene descriptions. MPEG does not prescribe or advise combinations of these profiles, but care has been taken that good matches exist among the different areas. MPEG-4 provides a large and rich set of tools for the coding of audiovisual objects. In order to allow effective implementations of the standard, subsets of the MPEG-4 Systems, Visual and Audio tool sets have been identified and can be used for specific applications. These subsets called profiles limit the total set a decoder has to implement. For each of these profiles, one or more levels have been set, restricting the computational complexity.

The visual part of the standard provides profiles for the coding of natural or synthetic/natural hybrid visual content. There are five profiles for natural video content:

- The simple visual profile provides efficient, error-resilient coding of rectangular video objects, which are suitable for applications on mobile networks.
- The simple scalable objects in the simple visual profile are useful for applications that provide services at more than one level of quality due to bit-rate or decoder source limitations, such as Internet use and software decoding.
- The core visual profile adds support for coding of arbitrary-shaped and temporally scalable objects to the simple visual profile. It is useful for applications such as those providing relatively simple content interactivity (Internet multimedia applications).
- The main visual profile adds support for coding of interlaced, semitransparent and sprite objects to the core visual profile. It is useful for interactive and entertainment-quality broadcast and DVD applications.
- The N-bit visual profile adds support for coding video objects having pixel-depths ranging from 4 to 12 bits to the core visual profile. It is suitable for use in surveillance applications.

The profiles for synthetic and synthetic/natural hybrid visual content are the following:

- The simple facial animation visual profile provides a simple means to animate a face model, suitable for applications such as audio-video presentation for the hearing impaired.
- The scalable texture visual profile provides spatial scalable coding of still image (texture) objects useful for applications needing multiple scalability levels, such as mapping texture on to objects in games and high-resolution digital still cameras.
- The basic animated 2D texture visual profile provides spatial scalability, SNR scalability, and mesh-based animation for still image (textures) objects and also to simple face object animation.
- The hybrid visual profile combines the ability to decode arbitrary-shaped and temporally scalable natural video objects with the ability to decode several synthetic

and hybrid objects, including simple face and animated still image objects. It is suitable for various content-rich multimedia applications.

Version 2 adds the following profiles for natural video:

- The Advanced Real-Time Simple (ARTS) profile provides advanced error-resilient coding techniques of rectangular video objects using a back channel and improved temporal resolution stability with the low buffering delay. It is suitable for real-time coding applications, such as the videophone, teleconferencing and remote observation.
- The core scalable profile adds support for coding of temporal and spatial scalable arbitrary shaped objects to the core profile. The main functionality of this profile is object-based SNR and spatial/temporal scalability for regions or objects of interest. It is useful for applications, such as the Internet, mobile and broadcast.
- The Advanced Coding Efficiency (ACE) profile improves the coding efficiency for both rectangular and arbitrary-shaped objects. It is suitable for applications such as mobile broadcast reception, the acquisition of image sequences and applications where high coding efficiency is requested.

The Version 2 profiles for synthetic and synthetic/natural hybrid visual content are the following:

- The advanced scalable texture profile supports decoding of arbitrary-shaped texture and still images, including scalable shape coding, wavelet tiling and error resilience. It is useful for applications that require fast random access as well as multiple scalability levels and arbitrary-shaped coding of still objects. Examples are fast content-based still image browsing on the Internet, multimedia-enabled Personal Digital Assistants (PDA) and Internet-ready high-resolution digital still cameras.
- The advanced core profile combines the ability to decode arbitrary-shaped video objects (as in the core visual profile) with the ability to decode arbitrary-shaped scalable still image objects (as in the advanced scalable texture profile). It is suitable for various content-rich multimedia applications such as interactive multimedia streaming across the Internet.
- The simple face and body animation profile is a superset of the simple face animation profile, obviously adding body animation.

Four audio profiles have been defined in MPEG-4 Version 1:

- The speech profiles provide Harmonic Vector Excitation Coding (HVXC), which uses a very low bit-rate parametric speech coder, a CELP narrow band/wide band speech coder and a Text-to-Speech Interface (TTSI).

- The synthetic profile provides score-driven synthesis and a TTSI to generate sound and speech at very low bit rates.
- The scalable profile, a superset of the speech profile, is suitable for scalable coding of speech and music for networks, such as Internet and Narrow Band Audio Digital Broadcasting (NADIB). The bit rates range from 6 Kb/s to 24 Kb/s, with bandwidth between 3.5 KHz and 9 KHz.
- The main profile is a rich superset of all other profiles, containing tools for natural and synthetic audio.

Another four profiles were added in MPEG-4 Version 2:

- The high quality audio profile contains the CELP speech coder and the low complexity AAC coder including long-term prediction. Optionally, the new Error-Resilient (ER) bit stream syntax may be used.
- The low-delay audio profile contains the HVXC and CELP speech coders, the low-delay AAC coder and the TTSI.
- The natural audio profile contains all natural audio coding tools available in MPEG-4, but not synthetic ones.
- The mobile audio internetworking profile contains the low-delay and scalable AAC object types, including twin VQ and Bit-Sliced Arithmetic Coding (BSAC).

Graphics profiles define which graphical and textural elements can be used in a scene. These profiles are defined in the Systems part of the standard [5.35]:

- The simple 2D graphics profile provides for only those graphics elements of the BIFS tool that are necessary to place one or more visual objects in a scene.
- The complete 2D graphics profile provides 2D graphics functionalities and support features, such as arbitrary 2D graphics and text, possibly in conjunction with visual objects.
- The complete graphics profile provides advanced graphical elements, such as elevation grids, and allows creating content with sophisticated lighting.

Scene description profiles allow audiovisual scenes with audio-only 2D, 3D or mixed 2D/3D content. The 3D profile is called VRML because it optimizes interworking with VRML material [5.35]:

- The audio-scene graph profile provides for a set of BIFS scene graph elements for use in audio-only applications. The audio-scene graph profile supports applications like broadcast radio.
- The simple 2D scene graph profile provides for only those BIFS scene graph elements necessary to place one or more audiovisual objects in a scene. The simple 2D scene

graph profile allows presentation of audiovisual content with potential update of the complete scene, but no interaction capabilities. The simple 2D scene graph profile supports applications like broadcast television.

- The complete 2D scene graph profile provides for all the 2D scene description elements of the BIFS tool. It supports features such as 2D transformations and alpha blending. The complete 2D scene graph profile enables 2D applications that require extensive and customized interactivity.
- The complete scene graph profile provides the complete set of scene graph elements of the BIFS tool. The complete scene graph profile enables applications like dynamic virtual 3D world and games.

The MPEG-J profile includes the following [5.35]:

- The Personal profile (a lightweight package for personal devices) addresses a range of constrained devices, including mobile and portable devices. Examples of such devices are cellular video phones, PDAs and personal gaming devices. This profile includes the following packages of MPEG-J APIs: network, scene and resource.
- The Main profile (includes all the MPEG-J APIs) addresses a range of consumer devices including entertainment devices. Examples of such devices are set-top boxes, computer-based multimedia systems and so forth. It is a superset of the Personal profile. Apart from the packages in the Personal profile, this profile includes the following packages of the MPEG-J APIs: decoder, decoder functionality, section filter and service information.

The Object Descriptor profile includes the following tools [5.8, 5.35]:

- Object Descriptor (OD) tool
- Synchronization Layer (SL) tool
- Object Content Information (OCI) tool
- IPMP tool

Currently, only one profile is defined that includes all these tools. The main reason for defining this profile is not to create a subset of tools, but rather to define levels for them. This applies especially to the SL because MPEG-4 allows multiple time bases to exist. In the context of levels for this profile, restrictions can be defined, for example, to allow only a single time base.

Verification Testing: Checking MPEG's Performance

MPEG carries out verification tests to check whether the standard delivers what it promises. The test results can be found on MPEG's home page [5.44].

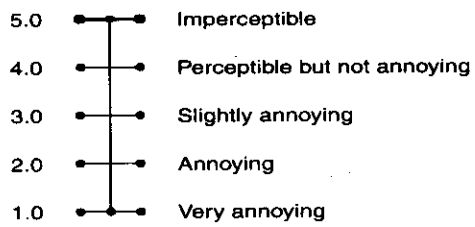

Figure 5.29 Subjective impairment scale for audio.

In video, a number of MPEG-4 capabilities have been formally evaluated using subjective tests. Coding efficiency, although not the only MPEG-4 functionality, is an important selling point of MPEG-4 and one that has been tested more thoroughly. Also, error robustness has been put to rigorous tests. Furthermore, scalability tests were done, and, for one specific profile, the temporal resolution stability was examined. Many of these tests address a specific profile.

MPEG-4 audio technology is composed of many coding tools. Verification tests have focused on small sets of coding tools that are appropriate in an application arena and hence can be effectively compared. Because compression is a critical capability in MPEG, the verification tests have, for the most part, compared coding tools operating at similar bit rates. The results of these tests are presented progressing from higher bit rates to lower bit rates. The exception of this is error robustness tools, the performance of which is noted at the end of this section.

The primary purpose of verification tests is to report the subjective quality of a coding tool operating at a specified bit rate. Most audio tests report this on the subjective impairment scale. This is a continuous five-point scale with subjective anchors as shown in Figure 5.29.

The error-robustness tools provide equivalently good error robustness across a wide range of channel error conditions and do so with only a modest overhead in bit rate. Verification test results suggest that the error-robustness tools used with an audio-coding system provide performance in error-prone channels that is nearly as good as the same coding system operating across a clear channel.

MPEG-4 Standardization Process

A standard must provide a minimum set of relevant tools, which, after being assembled according to industry needs, provide the maximum interoperability at a minimum complexity and lowest cost [5.45]. The success of the MPEG standard is mainly bounded by the "one functionality, one tool" principle. MPEG wants to offer the users interoperability and flexibility at the smallest complexity and cost. In order to fulfill these objectives, MPEG follows a development process with the following steps [5.45]:

1. Identify relevant applications using input from MPEG members.
2. Identify the functionalities needed by the applications determined in Step 1.
3. Describe the requirements following from the functionalities determined in Step 2 in such a way that common requirements can be identified for different applications.
4. Identify which requirements are common across the areas of interest and which are not common, but are still relevant.

5. Specify tools that support the requirements. Verify that the tools developed can be used to assemble the target systems and to provide the desired functionalities with an adequate level of performance. This is done by means of the so-called verification tests. The verification tests consist of formal subjective tests aimed at evaluating the quality of either audio or video signals processed using specific MPEG algorithms.

Two working tools play a major role in the development phase that follows the initial competitive phase: the working model and core experiments [5.46]. In MPEG-4, these phases are independent: VMs for the video, audio and synthetic and Natural Hybrid Coding (SNHC) and systems development.

Requirements for MPEG-4

The vision behind the MPEG-4 standard was explained through the eight new or improved functionalities described in the MPEG-4 Proposal Package Description (PPD). These eight functionalities came from an assessment of the functionalities that would be useful in future applications, but were not supported or not well supported by the available coding standards. The eight new or improved MPEG-4 functionalities formed the following three areas, the convergence of which MPEG-4 wanted to address [5.47].

- *Content-based interactivity*—This includes content-based multimedia data access tools, content-based manipulation and bit stream editing, hybrid natural and synthetic data coding, and improved temporal random access.
- *Compression*—This improves coding efficiency and coding of multiple concurrent data streams.
- *Universal access*—This includes robustness in an error-prone environment and content-based scalability.

Requirements for the MPEG-4 video standard are shown in Table 5.8.

Table 5.8 Requirements for the MPEG-4 Video standard [5.48].

Functionality	MPEG-4 Video Requirements
Content-based Interactivity	
Content-based manipulation and bit stream editing	This includes support for content-based manipulation and bit stream editing without the need for transcoding.
Hybrid natural and synthetic data coding	This includes support for combining synthetic scenes or objects with natural scenes or objects. This includes the ability for compositing synthetic data with ordinary video, allowing for interactivity.

Table 5.8 Requirements for the MPEG-4 Video standard [5.48]. (Continued)

Functionality	MPEG-4 Video Requirements
Improved temporal random access	This includes provisions for efficient methods to access parts randomly, within a limited time and with fine resolution, for example, video frames or arbitrarily shaped image content from video sequences. This includes conventional random access at very low bit rates.
Compression	
Improved coding efficiency	MPEG-4 Video shall provide subjectively better visual quality at comparable bit rates compared to existing or emerging standards.
Coding of multiple concurrent datastreams	This includes provisions to code multiple views of a scene efficiently. For stereoscopic video applications, MPEG-4 shall allow the ability to exploit redundancy in multiple viewing points of the same scene, permitting joint coding solutions that allow compatibility with normal video as well as the ones without compatibility constraints.
Universal Access	
Robustness in error-prone environments	This includes provisions for error-robustness capabilities to allow access to applications across a variety of wireless and wired networks and storage media. Sufficient error robustness shall be provided for low bit rate applications under severe error conditions (for example, long error bursts).
Content-based scalability	MPEG-4 shall provide the ability to achieve scalability with fine granularity in content, quality (for example, spatial and temporal resolutions) and complexity. In MPEG-4, these scalabilities are especially intended in content-based scaling of visual information.

5.5.2 MPEG-4 Systems

In MPEG-4, in addition to overall architecture, multiplexing and synchronization, the Systems part encompasses scene description interactivity, content description, and programmability. The combination of the exciting new ways of creating interactivity in audiovisual content offered by MPEG-4 Systems and the efficient representation tools provided by the Visual and Audio parts promise to be the foundation of a new way of thinking about audiovisual information. The goal

of specifying a standard way for the description and coding of audio-visual objects was the primary motivation behind the development of the tools in the MPEG-4 Systems [5.49, 5.50].

MPEG-4 Systems requirements may be categorized into two groups: traditional MPEG Systems requirements and specific MPEG-4 Systems requirements [5.48]. Traditional MPEG Systems requirements are streaming, synchronization and stream management. By streaming, we mean that the audiovisual data is to be transmitted piece by piece in order to match the delivery of the content to clients with limited network and terminal capabilities. In synchronization, the different components of an audiovisual presentation are closely related in time. For most applications, audio samples with associated video frames have to be presented together to the user at precise instances in time. In stream.management, the complete management of streams of audio visual information implies the need for certain mechanisms to allow an application to consume the content.

Specific MPEG-4 Systems requirements represent the ideas central to MPEG-4 and are completely new in MPEG Systems. The foundation of MPEG-4 is the coding of audiovisual objects. As for MPEG-4 terminology, an audiovisual object is the representation of a natural or synthetic object that has an audio and/or visual manifestation. Natural is generally understood to mean representations of the real world that are captured using cameras, microphones and so on, as opposed to synthetically generated material. The advantages of coding audiovisual objects can be summarized as follows:

- Allows interaction with the content
- Improves reusability and coding of the content
- Allows content-based scalability

In order to be able to use these audiovisual objects in a presentation, additional information needs to be transmitted to the clients' terminals. The individual audiovisual objects are only a part of the presentation structure that an author wants delivered to the consumers.

MPEG-4 Systems Architecture

The overall architecture of the MPEG-4 Systems is shown in Figure 5.30. Starting at the bottom, we first encounter the storage/transmission medium. This refers to the lower layers of the delivery infrastructure. Transportation of the MPEG-4 data uses a variety of delivery systems. This includes MPEG-2 TSs, UDP, ATM AAL2, MPEG-4 files or the Digital Audio Broadcasting (DAB) multiplexer. The FlexMux tool can optionally be used on top of the existing transport delivery layer. Regardless of the transport layer used and whether the FlexMux option is used, the delivery layer provides to the MPEG-4 terminal a number of ESs. In order to isolate the design of MPEG-4 from the specifics of the various delivery systems, the concept of the DMIF Application Interface (DAI) was defined [5.51]. This interface defines the process of exchanging information between the terminal and the delivery layer. The DAI defines procedures for initializing an MPEG-4 session and for obtaining access to the various ESs that are contained in it. These streams can contain a variety of different information: audiovisual object data scene

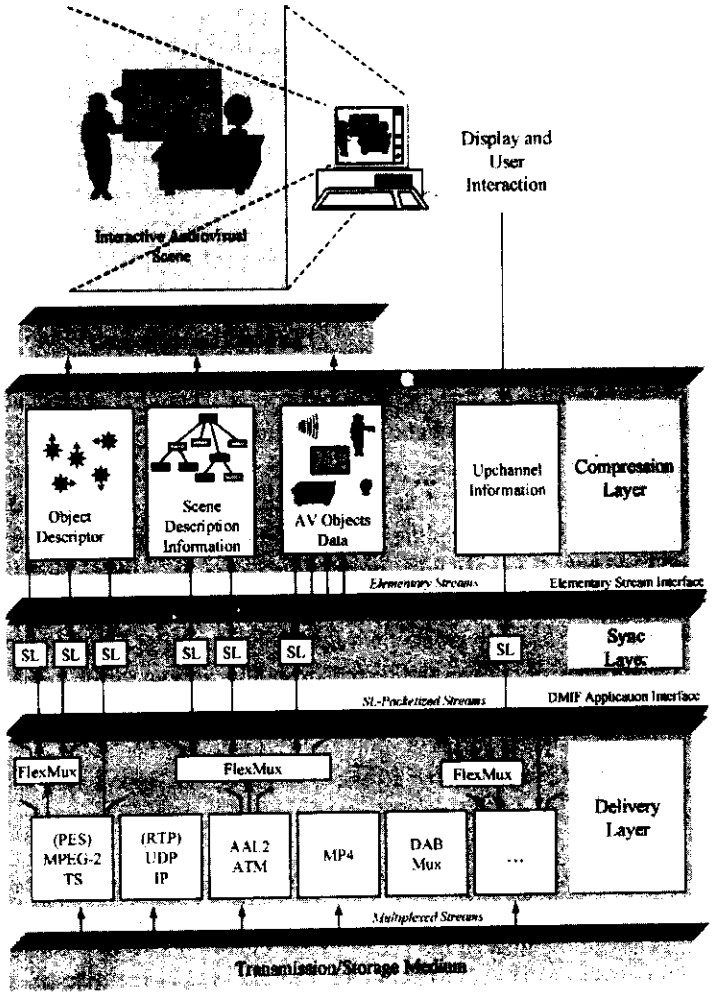

Figure 5.30 MPEG-4 Systems architecture. ©2000 ISO/IEC.

description information, control information in the form of object descriptors and metainformation that describes the content or associated intellectual property rights to it. The SL allows the inclusion of timing, fragmentation and continuity information on associated data packets. Such information is attached to data units that comprise complete presentation units, for example one entire VOP or an audio frame. These are called access units. An SL header contains no packet length indication. This is because it is assumed that the delivery layer that processes SL packets will already make such information available. The SL is the sole mechanism of implementing timing and synchronization mechanisms in MPEG-4. From the SL information, we can recover a time base as well ESs. The streams are sent to their respective decoders that process the data and that produce compensation units (for example, a decoded VOP).

In order for the receiver to know what type of information is contained in each stream, control information in the form of object descriptors is used. These descriptors associate sets of ESs to one audio or visual object, define a scene description stream, or even point to an object descriptor stream. These descriptors are the way with which a terminal can identify the content being delivered to it. The scene description information defines the spatial and temporal position of the various objects, their dynamic behavior and any interactivity features made available to the user. The scene description contains pointers to object descriptors when it refers to a particular audiovisual object. It is possible that an object (in particular synthetic objects like text and simple graphics) may be fully described by the scene description. As a result, it may not be possible to uniquely associate an audiovisual object with just one syntactic component of MPEG-4 Systems. The system's compositor uses the scene description information, together with decoded audiovisual object data, in order to render the final scene that is presented to the user. The scene description tools provide mechanisms to capture user or system events. In particular, they allow the association of events to user operations on desired objects that can, in turn, modify the behavior of the stream. The use of an object-based structure, where composition is performed at the receiver, considerably simplifies the content operation process. Starting from a set of coded audiovisual objects, it is very easy to define a scene description that combines these objects in a meaningful presentation. A similar approach is used in HTML and Web browsers, thus allowing even nonexpert users to create their own content easily. The fact that the content's structure survives the process of coding and distribution also allows for its reuse. For example, content filtering and/or searching applications can be easily implemented using ancillary information carried in object descriptors.

An audiovisual terminal contains three layers: Delivery Layer, Sync Layer and Compression Layer:

- The Delivery Layer encapsulates all the transport and multiplex functionality. It is mostly defined outside the MPEG-4 Systems standard. Some adaptation to (arbitrary) transport layers are defined in the scope of the DMIF, which is published as Part 6 of the MPEG-4 standard. The interface to the Delivery Layer is called the DAI. MPEG-4 Systems just specifies an optional multiplex tool, called FlexMux, that may be used in the adaptation to an existing or future transport layer. This structure provides significant flexibility in deploying MPEG-4 in a variety of communications environments.
- The Sync layer adapts ES data for communication across the DAI, providing timing and synchronization information, as well as fragmentation and random access information. The Sync Layer on the receiver side extracts this timing information to enable synchronized decoding and, subsequently, composition of the ES data. Its syntax is configurable and can also be empt
- The Compression Layer recovers data from its encoded format (ES) and performs the necessary operations to reconstruct the original information. It incorporates the

audiovisual object decoders. The decoded information is then used by the terminal's compositions, rendering and presentation subsystems.

Elementary Stream Management (ESM)

The term ESM is used to refer to the entire set of functionalities needed to describe, express relations between and affect synchronization among such datastreams. Within the extensive set of tools defined by MPEG-4, the ESM tools play a critical role in joining several building blocks together. ESM provides an alternative to the scene description language in that it limits the streaming resources of a presentation to the scene.

The ESM part of Systems also specifies means to identify and name ESs so that they can be referred to in a scene description and can be attached to individual objects. This association is performed in object descriptors that are transmitted in their own ESs. Object descriptors are separate from the scene description itself, thus simplifying editing and remultiplexing of MPEG-4 content. The descriptors associated with audiovisual objects are nodes in the scene related to ES identifiers. An additional mapping is required to resolve these identifiers to actual transport layer channels (for example, port numbers). How this mapping is performed depends on the Delivery Layer instance that is actually used. In accordance with the goal of allowing the use of any Delivery Layer, MPEG-4 does not define this mapping, but rather expects parties that define these delivery layers to define how MPEG-4 content should be mapped to their design in a way that they consider most appropriate.

A hierarchically structured set of descriptors constitutes the major building block of the object description framework. The highest-level descriptor is the OD itself. It serves as a shell that aggregates a number of other descriptors. Most prominent among these are the ES descriptors which describe individual streams. Additional auxiliary information can be attached to an OD, either to describe the content conveyed by these streams in a textual form object content information, or to do IPMP.

Main components of the OD are depicted in Figure 5.31. The linking to the scene is achieved in a two-stage process. First, there is a numeric identifier, called the object descriptor (OD_ID), that labels the object descriptor. This identifier is referenced by the scene description stream and thus links the OD to the scene. The second stage involves the actual binding of ESs identified and described by the ES descriptors included in this OD, using another identifier, which is part of the ES descriptor.

Example 5.10 In the most simple case, an OD contains just one ES descriptor that identifies the audio stream that belongs to the audio source node by which this OD is referenced, as illustrated in Figure 5.32a. The same OD may as well be referenced from two distinct scene description nodes as shown in Figure 5.32b.

Example 5.11 Within a single OD, it is possible to have two or more ES descriptors: one identifying a low bit rate audio stream and another identifying a higher bit rate stream with the same content (Figure 5.33a). In this case, the terminal, or rather the user, has a choice between

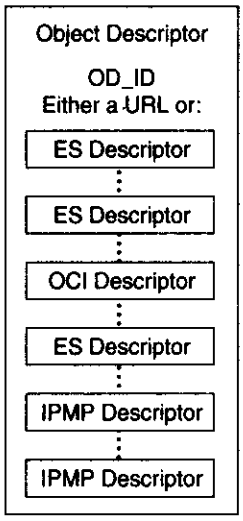

Figure 5.31 Main components of the OD. ©2000 ISO/IEC.

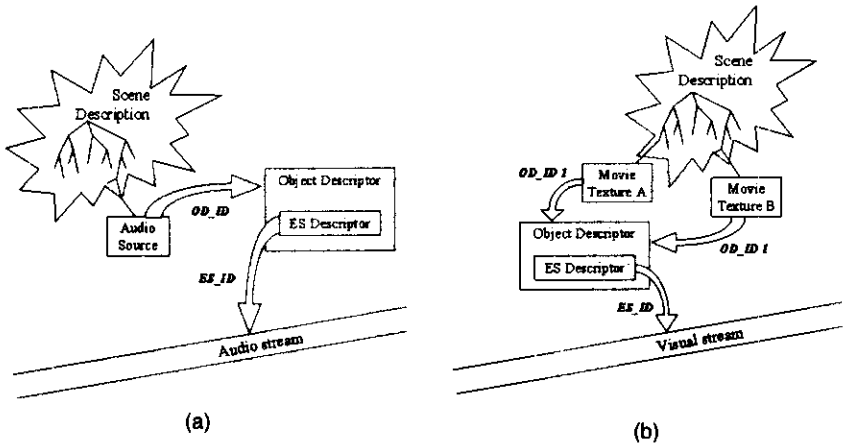

Figure 5.32 ES reference through OD (a) from one scene description node and (b) from two scene description nodes.©2000 ISO/IEC.

two audio qualities. Specifically, for audio, it is also possible to have multiple audio streams with different languages that can be selected according to user preferences (Figure 5.33b).

It is possible to describe within one OD a set of streams that corresponds to a scalable or hierarchical encoding of the data that represents an audiovisual object. In that case, it is necessary to signal not only the properties of the individual ES, but also their interdependencies. Simple and more complex ES dependence indication is shown in Figure 5.34. In Figure 5.34a, each stream depends on the previous one, but, in Figure 5.34b, the same base layer stream is referenced by two other streams, providing, for example, quality improvements in the temporal domain (temporal scalability) and in the spatial domain (spatial scalability). The precise instruc-

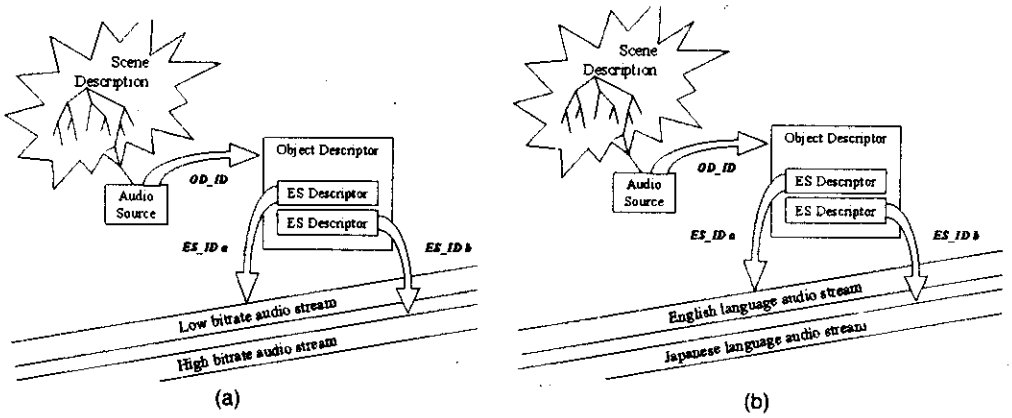

Figure 5.33 Reference to multiple ESs though one OD (a) and audio application for multiple languages (b). ©2000 ISO/IEC.

tions on how the decoder has to handle each individual stream in such a case is incorporated in a decoder-specific info subdescriptor that is contained in each ES descriptor.

ODs are the glue between the scene description and the ESs. An ES descriptor (a part of an OD) is associated with each ES and contains all the information needed to find it, describe it and advise the receiver which resources need to be set up to decode it. ODs are transported in an ES of their own. They are never sent just as is, but are always encapsulated in so-called OD commands.

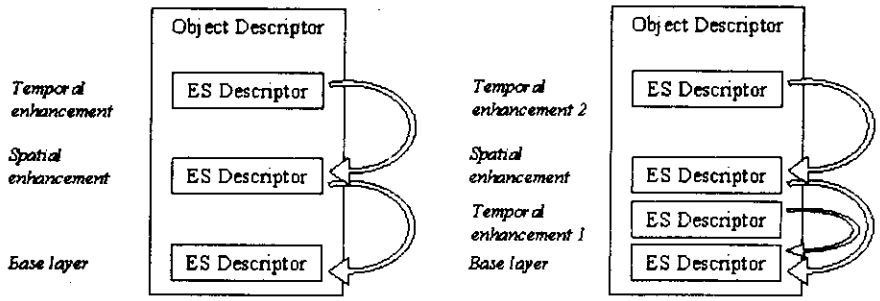

Figure 5.34 Simple (a) and more complex ES dependence indication (b). ©2000 ISO/IEC.

Auxiliary Descriptors and Streams

An OD points to auxiliary streams associated to this set of media streams. There are three types of such streams. First, semantic information about the content of an audiovisual object that is fed by these streams may be covered by means of OCI. Then, IPMP information may be attached. Finally,

clock-reference streams may be used to convey time-base-information. Another type of auxiliary information is the QoS descriptors, which may be attached to ES descriptors. The design of the descriptors as self-describing entities constitutes a generic extension mechanism that may be used by ISO or specific applications to attach arbitrary descriptive information to an audiovisual object.

Object content information consists of a set of OCI descriptors that communicate a number of features of the audiovisual object that is constructed by the ESs associated with a given OD. There is a descriptor with keywords, possibly to be used with search engines, a textual description of the content, language and content rating information. These descriptors may be included directly in ODs to indicate static OCI properties. An OD for an audio object may contain a set of different language streams that is activated according to user preferences. In case of OCI changes over the lifetime of the media streams associated with this OD, it is possible to attach an OCI stream to the OD. The OCI stream conveys a set of OCI events that are qualified by their start time and duration. This means that an MPEG-4 presentation may be further decomposed semantically based on the actual content presented, which includes OCI streams as well as scene description streams. This influences their scope. As an example, attaching an OCI stream to a media stream as well as to a whole subscene is presented in Figure 5.35.

The IPMP framework consists of a fuliy standardized Intellectual Property Identification (IPI) descriptor as well as IPMP descriptors and IPMP streams with a standardized shell and not with a standardized content. The core IPMP framework consists of IPMP descriptors and IPMP streams.

IPMP framework has been engineered in a way that should allow the coexistence of multiple IPMP systems that govern the conditional access to specific content streams or an entire presentation. IPMP descriptors convey proprietary information that may help to decrypt ESs or that contain authorization or entitlement information evaluated by unequal proprietary IPMP subsystems in the receiving terminal. IPMP descriptors may occasionally be changed over time.

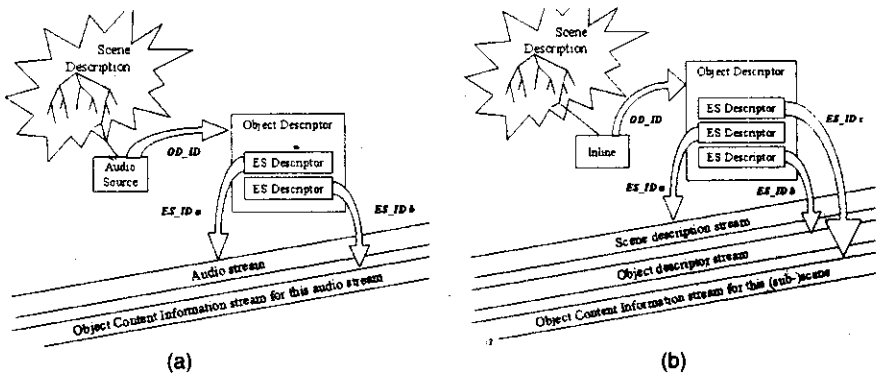

Figure 5.35 Attaching an OCI stream to (a) a media stream or to (b) a whole subscene. ©2000 ISO/IEC.

However, IPMP information that needs frequent updates in a streaming fashion should be conveyed in IPMP streams. IPMP streams also keep IPMP information separate from the original MPEG-4 information.

The QoS descriptor is an additional descriptor that may occur in ES descriptors. It aims to specify the requirements that a specific ES has on the QoS of the transport channel for this stream. The most obvious QoS parameter is the stream priority and a possibility to signal predefined QoS scenarios, most notably guaranteed and best effort channels. QoS descriptors at the ES level have an obvious use in interactive scenarios where the receiving terminal may select individual ESs based on their traffic and other QoS requirements as well as the associated communication cost [5.52].

Because MPEG-4 defines audiovisual objects rather than programs, ISO/IEC decided to rename the "program clock reference" to an object clock reference. An object clock reference is a sample from an object time base. Different objects within the same presentation may have different object time bases, even though this may make tight synchronization impossible [5.53].

Structuring Content by Grouping of Streams

An interesting question is how to use grouping of ESs through one OD to improve the structure of the content. In particular, we want to be able to arrange a large number of elementary streams into groups. Even though initial MPEG-4 applications may only require a small number of streams, the design allows efficient management of content that consists of a large number of streams. As hardware and software capabilities improve, it is only natural that the sophistication of content will also increase.

A number of application scenarios are conceivable and could include the delivery of differentiated quality content to different user groups, delivery of different portions of the content to different user groups and reception of content originating from different sources. Grouping is quite relevant for multicast applications where it should be easy to remove parts of the content somewhere on the way, but in point-to-point applications, it should also be possible to negotiate the desired subset of ESs between client and server.

Managing Content Complexity

Initial ODs convey some indication of profiles and levels of the content referenced by them. These scalar indications allow an immediate decision by the receiving terminal about whether it is able to decode and present the content being received. Due to the potential complexity in terms of the number of scenes, it has also been made possible to indicate such complexity only for the current subscene, that is, excluding parts of the scene that are included through inline nodes [5.52].

In the absence of profile and level indications or at the discretion of the receiving terminal, it is also possible to evaluate the decoder configuration, stream priorities, bit-rate requirements and the dependencies signaled in the ES descriptors. This allows the receiving terminal in an interactive application to request the delivery of a meaningful subset of streams for each media object so that the computational resources of the terminal are not exceeded. Apart from

these resource-driven considerations, the terminal or the user needs to evaluate the scene description in order to decide which subset of the media objects advertised in an OD stream are relevant.

Distributed Content-Handling Considerations

MPEG-4 allows the construction of content where different parts may originate from different locations. Furthermore, MPEG-4 makes no assumptions about the type of the underlying communications infrastructure (IP, ATM, broadcast, and so forth). It is then impossible to ensure that use of MPEG-4's distributed content capabilities will result in seamless content presentation in all circumstances.

We expect that content creators, jointly with both content and communication services providers, will create content cognizant of the environment in which it will operate, leveraging the available resources in order to provide a maximum-quality user experience. This is similar to the practice of tuning HTML pages so that they provide both visually rich content as well as reasonable download times across telephone lines. At any rate, the expectation is that the evolution of the Internet and other suitable transport networks will gradually solve these issues [5.52].

System Decoder Model (SDM) for ES Synchronization

An SDM is used to specify the behavior of a receiving MPEG-4 terminal in terms of a timing model and a buffer model. The SDM receives individual ESs from the Delivery Layer through the DAI. MPEG-4 requirements on the end-to-end delivery of data through the DAI, most prominently, a constant end-to-end delay [5.54], are specified. The SDM is shown in Figure 5.36 [5.52]. It consists of the DAI, a number of buffer decoders, composition memories and the compositor. The coder entity for the purposes of the SDM is the Access Unit (AU). Each ES is partitioned in a sequence of such AUs. The AU is the smallest entity to which timing information can be associated. The Sync Layer defines some syntax (the SL packet headers) that carries timing information, that is, time stamps and clock references. This data allows a receiver to determine which portions of different streams are to be composed and, hence, presented at the same time. The syntax of the SL encodes both AU boundaries and the timing information associated with AUs.

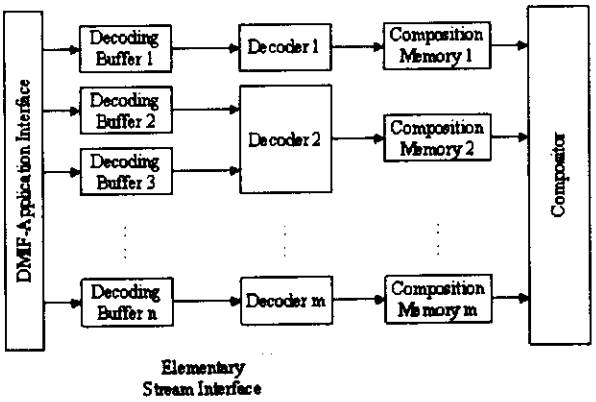

Figure 5.36 SDM. ©2000 ISO/IEC.

The DAI supplies AUs or parts thereof to the decoding buffer that stores the AUs until their decoding time. At that point in time, the SDM assumes instantaneous decoding of the AU, removal of the AU data from the decoding buffer and appearance of the decoded data corresponding to the AU in the associated composition memory. With this model, it is possible for the encoding terminal to know how many decoding buffer resources are available in the receiving terminal for a specific stream at any point in time. The content of decoding buffers is consumed by decoders. In the case of hierarchically encoded audio-video objects, a decoder may be connected to multiple decoding buffers as in the case of Decoder 2 in Figure 5.36. A decoder outputs the decoded data to one composition memory. The decoded data is grouped in composition units. The relation between AUs and composition units is assumed to be known for each specific decoder type. Each composition unit of decoded data is available for composition starting at an indicated composition time, either known implicitly or through an explicit composition time of the subsequent composition unit. Only a minimum of assumptions is made on the compositor for the purpose of defining the SDM. This includes that the compositor instantaneously samples the content of each composition memory of the MPEG-4 terminal.

MPEG-4 Systems BIFS

BIFS provides a complete framework for the presentation engine of an MPEG-4 terminal. It expresses how individual audiovisual objects are to be composed together for presentation on the user's screen and speakers.

As a general paradigm in MPEG-4, all information is conveyed in a streaming manner. The term "ES" refers to data that fully or partially contains the encoded representation of a single audio or visual object, scene description information or control information.

ESs or groups therefore are identified and characterized by ODs. These include the scene description, audiovisual object data and OD streams themselves. The information contained will include the format of the data, for example, a visual face animation stream. Also, the indication of the resources necessary for decoding for example, a profile/level indication, will be included. Alternate representation or scalable encoding using multiple streams for a single audiovisual object can also be signaled by the ODs. For example, an object descriptor may specify a set of elementary streams that jointly contain the compressed representation of an audio signal.

BIFS enables mixing various MPEG-4 media together with 2D and 3D graphics, handling interactivity, and dealing with the local or remote changes of the scene over time.

BIFS is actually composed of four elements:

- The operational elements of the scene, consisting of nodes and routes. These represent the following in particular:
  - Audiovisual objects and their attributes (which define their audiovisual properties)
  - Composition operations
  - Animation of the content
  - Interactive behavior of individual objects by linking event fields to event sink fields between different nodes

- The binary syntax for compressing the node tree as well as the associated routes
- The BIFS-Command protocol, in order to stream scene changes, insert new scenes or objects, delete objects and so forth
- The BIFS-Animation protocol, in order to stream animations of node parameters which is used as a very low overhead mechanism to animate audiovisual objects

A central concept in the MPEG-4 design is transmission and interaction with audiovisual objects of synthetic or natural nature. The Audio and Visual parts of the standard provide the encoding algorithms for individual audiovisual objects. In order to combine these media together into complete presentations, a scene description capability is needed. BIFS provides the input data to the presentation layer of the MPEG-4 terminal. No other scene format covers all the requirements of the MPEG-4 presentation engine. The main concepts driving the design of the BIFS specification are the following:

- *Integration of 2D and 3D synthetic media together in a single format*—By avoiding the burden of mixing multiple media formats together, the content creator has a way to design a complete multimedia content without the hassle of dealing with many different formats, and the end-users benefit from a lighter terminal with state-of-the-art media capabilities.
- *Streaming environment*—All existing scene description formats are designed so that a complete scene has to be downloaded before anything can be viewed on the terminal. In MPEG-4, the terminal is linked to one or several MPEG-4 servers. The scene description, as with any other media, has to be streamed to the client. Allowing the scene to be cut into pieces and streamed to the client, as well as its animation parameters, provides more communication applications. These streaming features are also necessary in order to send new data to the user during the communication.
- *Compression*—Most existing scene representation is in text format, making it editable, but very inefficiently represented in terms of data size. The sizes of scenes are often much smaller than other media. However, complex scenes can be large datasets, sometimes up to several megabytes. Even for smaller scenes, being able to reduce the data size can bring significant improvements in transmission time, especially at low bit rates or in broadcast environments in which the scene has to be transmitted repeatedly. Moreover, animation data can also be streamed in MPEG-4. The efficient compression of such data can significantly reduce the bit rates. A typical example is for a BIFS animation stream that consumes 10 Kb/s, noncompressed data would consume more than 120 Kb/s.

Scene description and stream description are separated strictly in MPEG-4. In particular, the scene description contains no information about the streams that are needed to reconstruct a particular audiovisual object. On the other hand, the stream description contains no information that relates to how an object is to be used within a scene. The scene description is usually con-

sciously authored by the content creator, but the stream description mostly follows from general content creator references, the default setting of an editing tool or even the service provider constraints and rules. The link between the two descriptions is a numeric OD identifier that the scene description uses to point to ODs, which, in turn, provide the necessary information for assembling the ES data required to decode and reconstruct the object at hand.

MPEG-4 specifies a BIFS that is used to describe scene composition information which includes the spatial and temporal locations of objects in scenes. Elements of the scene and relationships between them form the scene graph that must be coded for transmission [5.49]. The scene graph is transmitted at the beginning of the content session and may be dynamically updated as the content plays with a special streams of BIFS update commands. Content developers have wide flexibility to use BIFS in a variety of ways [5.55]. Audio BIFS, like the rest of BIFS, is composed of a number of nodes that can be interlinked to form a scene graph. However, the concept of the audio BIFS scene graph is somewhat different. It is termed an audio subgraph. Although the main (visual) scene graph represents the position and orientation of visual objects in presentation space and their properties such as color, texture and layering, an audio subgraph represents a signal flow graph describing digital signal-processing manipulation.

Designed as a file format for describing 3D models and scenes ("worlds" in VRML terminology), VRML faces some important features that are required for the types of multimedia applications targeted by MPEG-4. In particular, the support for natural video and audio is basic, but the timing model is loosely specified, implying that synchronization in a scene composed of multiple media types cannot be guaranteed. VRML was proposed and developed by the VRML Consortium. Their VRML 2.0 specification became an ISO/IEC international standard in 1998 [5.56].

Both VRML and MPEG-4 BIFS rely on the scene graph to describe the organization of audiovisual material. A scene graph represents a set of hierarchically related nodes. Each node in the visual scene graph represents a visual object (like a cube or image), a property of an object (like the textural appearance of a face or a cube) or a transformation of part of the scene (like a rotation or scaling operation). By connecting multiple nodes together, object-based hierarchies are formed. For example, one node might correspond to the location of a virtual character. The subgraphs, or sets of connected nodes, would represent the head and limbs of a character. By transforming the positions of the limbs, they may be made to move. By transforming the position of the character, all of the subgraphs are automatically transformed as well. Therefore, the character moves, but the limbs stay in the same relative positions. Each node has several fields that detail the properties of the object. For an object node like a cube, the fields give the size and shape of the object. For a property mode, the fields specify particular properties, such as the color of the cube and the image to be texture–mapped to the cube. For a transform node, the fields specify the set of subsidiary nodes that are affected by the transformation, as well as the details of the transform.

BIFS is a binary format, but VRML is a textual format. This is a fundamental difference between the two. Thus, although it is possible to design scenes that are compatible with both

BIFS and VRML, transcoding of the representation formats is required. In what follows, we highlight the functionalities that BIFS adds to the basic VRML set [5.56].

BIFS describes an efficient binary representation of the scene graph information. The coding may be either lossless or lossy. The coding efficiency is derived from a number of classical compression techniques, plus some novel ones. The technique is based on the fact that, given some scene graph data that has been previously received, it is possible to anticipate the type and format of data to be received subsequently.

Example 5.12 The use of context in efficient coding is described in the following listing. Quantization of numerical values is supported, as well as the compression of 2D meshes as specified in MPEG-4 Visual [5.57].

```
Consider a simple coding scheme with the following tags:
            <begin>       - beginning of record
            <end>     - end of record
            <break>       - end of element in record
            <string>      - text string follows
            <number> - number follows
We wish to use this scheme code for a record consisting of first name, last
name and phone number, for example:
            First name: Jim
            Last name: Brown
            Phone:        7771234
With no knowledge context we would need to code this as:
<start><string>Jim<break><string>Brown<break><number>7771234<break><end>
If the context is known, i.e. we know that the structure of the record is
"string, string, number", we do not have to spend bits specifying the type
of each element:
<start>Jim<break>Brown<break>7771234<end>
```

BIFS is designed so that the scene may be transmitted as an initial scene followed by time-stamp modifications to the scene. For dynamic scenes that change over time, this leads to a huge improvement in memory use and reduced latency when compared to equivalent VRML scenes. The BIFS command protocol allows replacement of the entire scene, addition, deletion and replacement of nodes and behavioral elements in the scene graph and modification of scene properties.

A second streaming protocol, BIFS animation, is designed to provide a low-overhead mechanism for the continuous animation of changes to numeric values of the components in the scene. These streamed animations provide an alternative to the interpolator nodes supported in both BIFS and VRML. The main difference is that interpolator nodes contain very large amounts of data that must be loaded in the scene and stored in memory. By streaming these animations, the amount of data that must be held in memory is reduced significantly. Secondly, by removing these data from the scene graph that must be initially loaded, the amount of data that must be processed in order to begin presenting the scene is also reduced.

BIFS includes native support for 2D scenes. This facilitates content creators who want to produce low-complexity scenes, including the traditional television and multimedia industries. Many applications cannot bear the cost of requiring decoders to have full 3D rendering and navigation. This is particularly true where hardware decoders must be of low cost. However, rather than simply partitioning the multimedia world into 2D and 3D, MPEG-4 BIFS allows the combination of 2D and 3D elements in a single scene.

VRML provides simple audio support. This support has been enhanced in MPEG-4. BIFS provides the notion of an audio scene graph, where the audio sources, including streaming ones, can be mixed. It also provides nodes to interlace to the various MPEG-4 audio objects [5.58]. Audio content can even be processed and transformed with special procedural code to produce various sound effects.

BIFS provides support at the scene level for the MPEG-4 facial animation decoder [5.57]. A special set of BIFS nodes exposes the properties of the animated face at the scene level. A full participant of the scene can be integrated with all BIFS functionalities, similarly to any other audio or visual objects.

As MPEG-4 Version 1 augments the virtual reality model of sound in VRML with a versatile abstract-effects model, MPEG-4 Version 2 extends the simple virtual reality model to include two rich and robust techniques for creating virtual audio environments. The first technique is physical, the second one is perceptual. By physical modeling of acoustic environments, we mean processing sound so that the acoustic effects-processing corresponds to the visual scene. Modeling of the acoustic environments is bound to the physical reality defined by the visual scene. Audiovisual interaction is one of the important features of virtual acoustics. The aim is a virtual environment where auditory and visual events are related. Audiovisual objects change both the auditory and visual characteristics, according to their position, orientation, materials and visibility in a scene. The source model in a virtual acoustic environment includes the sound content and directivity properties of the emitter, which can be modeled efficiently using digital filters. The environment model aims at reproducing the effects of the surrounding space (listening room, concert hall, metro station, and so forth). The most efficient approaches are time-domain hybrid methods combining ray tracing and an image source method for direct sound and early reflections with late reverberation modeling based on statistical parameters [5.59]. The listener model is closely related to the method of reproducing the auditory sensation. Different 3D processings are needed for different types of reproduction like headphone, stereophonic and multichannel loudspeaker listening.

In perceptual techniques, creation and modification of environment sound characteristics are based upon perceptual parameterization. Perceptual parameters have recently been introduced into the MPEG-4 Version 2 as another method of creating environmental acoustic effects in the scene, independent of the visual (and physical) reality. The creation of environmental acoustic effects is enabled separately for each sound source, together with characterization of the perceptual quality of the source and the environment in a 3D space. High-level perceptual parameters (source presence and brilliance, room reverberance, heaviness, liveness, and envel-

opment) are used to derive low-level energy parameters for the control of direct sound and the different parts of the room impulse response, that is, the directional and diffuse early reflections, as well as the late reverberation [5.59, 5.60, 5.61]. The high-level parameters have been derived based on subjective testing of perceived room acoustic quality. Based on these parameters, a real-time spatial sound-processing scheme has been derived [5.61]. This enables computationally efficient, yet perceptually relevant, 3D audio rendering.

The scene description can be dynamically changed at any time. An initial scene description is provided at the beginning of an MPEG-4 stream. It can be as simple as a single node or as complex as one wants (within limits that are established for ensuring conformance). BIFS commands are used to modify a set of properties of the scene at a given time. It is possible to insert, delete and replace nodes, fields and routes as well as to replace the entire scene. For continuous changes of the parameters of the scene, BIFS animation can be used; it specifically addresses the continuous update of the fields of a particular node. BIFS animation is used to integrate different kinds of animation, including the ability to animate face models as well as meshes, 2D and 3D positions, rotations, scale factors and color attributes. The BIFS animation information is conveyed in its own elementary stream.

5.5.3 DMIF

DMIF is a session protocol for the management of multimedia through generic delivery technologies [5.35]. Different from its predecessor, MPEG-4 has been targeted since the beginning to adopt to multiple operating scenarios (local retrieval, remote interaction, broadcast or multicast) and delivery technologies. The design choice was to abstract the functionality that the delivery layer has to provide and to focus the MPEG-4 Systems activity on the common features. The goal is still to produce effective solutions. However, a demarcation has been drawn between the aspects that can be managed uniformly and independently from the aspects that can be managed uniformly and independently from the delivery technology (included in MPEG-4 Systems), and those that instead are impacted by the delivery technology and by the operating scenario (included in MPEG-4 DMIF). This demarcation line is named DAI. The idea is shown in Figure 5.37 along with the generic MPEG-4 layered model, which comprises the Compression Layer, the Sync Layer (that is part of systems) and the Delivery Layer (DMIF) [5.54].

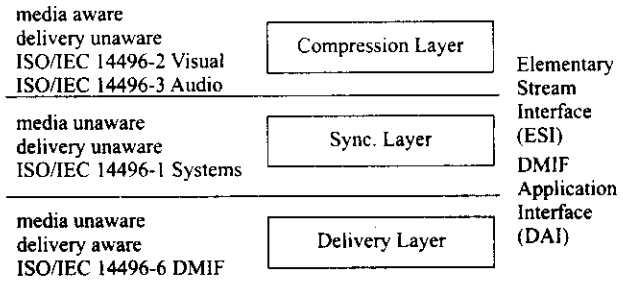

Figure 5.37 MPEG-4 layered model.

The Compression Layer performs media encoding and decoding of ESs [5.62, 5.63]. The Sync Layer manages ESs and their synchronization and hierarchical relations [5.64]. The Delivery Layer ensures transparent access to content irrespective of delivery technology [5.51]. The functionality provided by DMIF is expressed by an interface called DAI and is transmitted into protocol messages. These protocol messages may differ based on the network on which they operate. The QoS is also considered in the DMIF design, and the DAI allows the DMIF user to specify the requirements for the desired stream. The DMIF specification provides hints on how to perform tasks on a few network types, such as the Internet. The DAI is also used for succeeding broadcast material and local files. This means that a single, uniform interface is defined to access multimedia contents on a multitude of delivery technologies. It is appropriate to state that the integration framework of DMIF covers three major technologies: interactive network technology, broadcast technology and storage technology. This is shown in Figure 5.38. Fulfilling the previous requirements represents an improvement to favor the development of truly multimedia applications. The identification of a common interface, from an application's point of view, to today's and tomorrow's networks, encompassing both the QoS-enabled networks and the best-effort model, is the first DMIF requirement. This is particularly useful in relation to QoS issues.

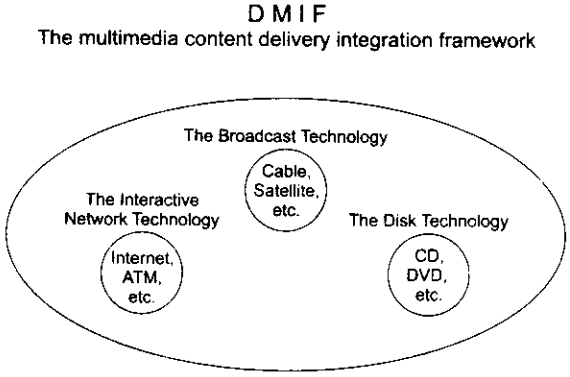

Figure 5.38 The delivery integration of three major technologies [5.36]. ©2001 ISO/IEC.

The second, less obvious requirement of the DMIF reference architecture is to also hide the operational scenario details to the application. It means managing the access to locally or remotely retrieved streams, as well as broadcast and multicast streams, through a common interface to the delivery system. The reasons for these requirements are not that evident, because broadcast content is certainly designed with different criteria than content meant to be retrieved interactively.

The DMIF communication architecture is represented in Figure 5.39. The picture shows how the different operational scenarios are uniformly modeled, through the identification of four basic blocks: originating application, originating DMIF, target DMIF and target application. The elements in the user part are meant to be part of the originating terminal, and the elements in the bottom are part of the target terminal in the case of a remote interactive scenario. The originating

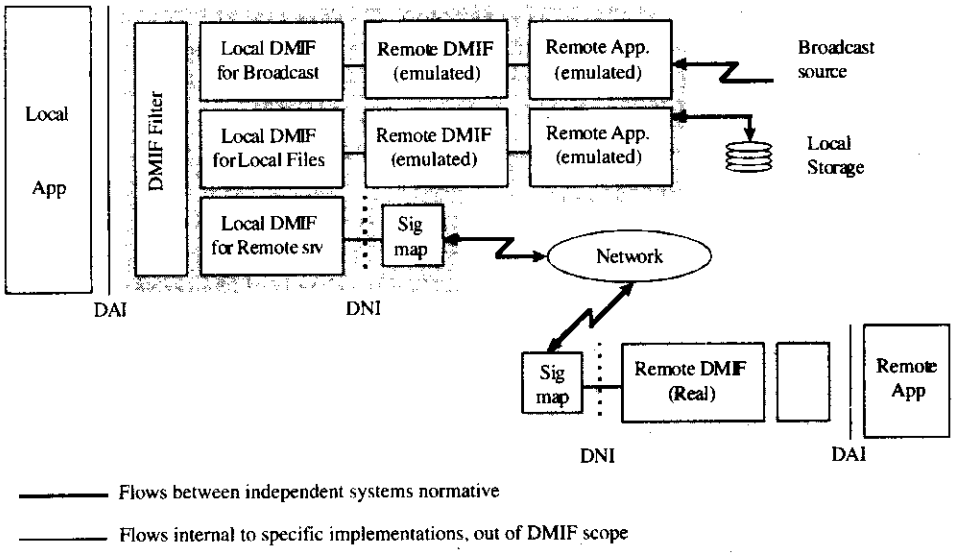

Figure 5.39 DMIF communication architecture [5.36]. ©2001 ISO/IEC.

DMIF module is meant to work in cooperation with the target DMIF module to provide a session-level service. The distinction between originating and target DMIF modules in the local retrieval and broadcast scenario is a bit artificial, but has been left for the uniformity with the remote interactive scenarios

The originating application is the actual application in the terminal, for example an MPEG-4 browser or a multimedia-conferencing application. It is assumed that it has, in all cases, a counterpart, the target application. The originating application interacts with the target application through DMIF. In the case of remote interactive operational scenarios, the two applications reside on separate hosts, and the communication between the two is regulated by some signaling protocol, not known by the applications themselves. The DMIF filter module is identified in the DMIF specification to highlight the potential benefit of the architecture. It represents a sort of container for the various DMIF instances available in a terminal, and its role is to select the appropriate DMIF instance to provide a certain service. Another module is the signaling module (Sig map). This element applies only to the DMIF instances for remote interactive scenarios and is kept separate from the other DMIF elements in order to highlight the role of the DMIF Network Interface (DNI). Particularly for the case of remote interactive systems, DMIF factorizes features that are instead specific for a certain network (such as signaling protocols). The DNI represents the border between the generic and specific tasks of a DMIF instance for remote interactive scenarios.

The DAI is a semantic API that derives directly from the requirement of hiding the delivery technology and operational scenario details from the application. Thus, local and remote retrieval are not different from multicast or broadcast. An MPEG-4 browser would be able to

access and present multimedia content uniformly and independently of the operational scenario. In the MPEG-4 context, the DAI formalizes the demarcation line between Systems and DMIF, and it separates the elements and tools of MPEG-4 that are conceptually network unaware from those that instead relate with the delivery technology (covered by DMIF).

DMIF allows the concurrent presence of one or more DMIF instances, each one targeted for a particular delivery technology, in order to provide support in the same terminal multiple delivery technologies and even multiple scenarios (broadcast, local storage, and remote interactive). Multiple delivery technologies may be activated by the same application, which could therefore seamlessly manage data sent by broadcast networks, local file systems and remote interactive peers.

DMIF Computational Model

When an application requests the activation of a service, it uses the service primitives of the DAI and creates a service session. The DMIF implementation then contacts its corresponding peer (that conceptually can be either a remote peer or a local emulated peer) and creates a network session with it. Network sessions have network-wide significance, and service sessions instead have local meaning. The acquisition between them is maintained by the DMIF Layer. In the case of broadcast and local storage scenarios, the way that the network session is created and then managed is out of the scope of this specification. In the case of a remote interactive scenario instead, DMIF uses the native signaling mechanism for that network to create and then manage the network session, for example, ATM signaling. The application peers then use this session to create connections that are used to transport application data, for example, MPEG-4 ESs.

When an application needs a channel, it uses the channel primitives of the DAI. DMIF translates these requests into connection requests that are specific to the particular network implementation. In the case of broadcast and local storage scenarios, the way that the connections are created and then managed is out of the scope of this specification. In the case of a networked scenario instead, DMIF uses the native signaling mechanisms for that network to create those connections. The application then uses these connections to deliver the service.

Figure 5.40 provides a high-level view of a service activation and the beginning of data exchange. The high-level walkthrough consists of the following steps in which DMIF is involved. The originating application requests the activation of a service to its local DMIF instance. A communication path between the originating application and the originating DMIF

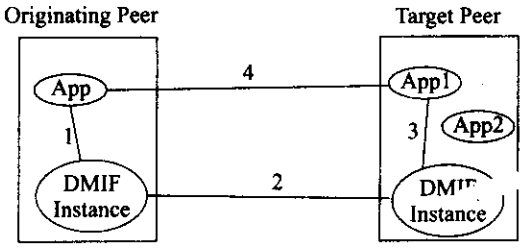

Figure 5.40 DMIF computational model [5.36]. ©2001 ISO/IEC.

instance is established in the control plane (1) and is associated to a locally meaningful service session.

The originating DMIF instance contacts the target DMIF instance. A communication path between these two is established in the control plane (2) and is associated with a unique network session.

The target DMIF instance identifies the target application and forwards the service activation request. A communication path between the target DMIF instance and the target application is established in the control plane (3) and is associated with a locally meaningful service session.

The peer applications create channels (requests flowing through communication paths 1, 2 and 3). The resulting channels in the user plane (4) will carry the actual data exchanged by the applications.

The DMIF specification defines an architecture that is open to future evaluations in the delivery technology and that is able, if actually implemented in the terminals, to protect investments in the development of multimedia applications.

5.5.4 MPEG-4 Video

Digital video is replacing analog video in many existing applications. A prime example is the introduction of Digital Television (DTV). Another example is the progressive replacement of analog video cassettes by DVD as the preferred medium to watch movies. MPEG-2 has been one of the key technologies that enabled the acceptance of these new media. In these applications, digital video will initially provide functionalities similar to analog video, that is the content is represented in digital form instead of analog with direct benefits, such as improved quality and reliability, but the content remains the same to the user. However, after the content is in the digital domain, new functionalities can easily be added. This allows the user to view, access and manipulate the content in completely new ways. The MPEG-4 standard provides key technologies that will enable such functionality.

The MPEG-4 Visual standard consists of a set of tools that enable applications by supporting several classes of functionalities. The most important features covered by the MPEG-4 standard can be summarized in three categories, as shown in Figure 5.41.

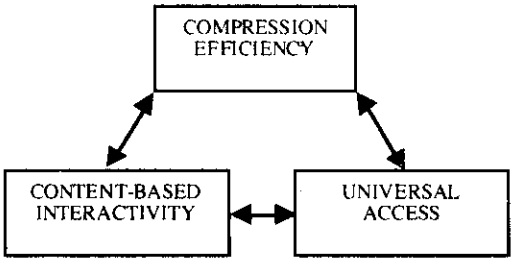

Figure 5.41 Functionalities in MPEG-4 Visual standard.

Compression efficiency has enabled applications such as DTV and DVD. Improved compression efficiency and coding of multiple data streams will increase the acceptance of applications based on MPEG-4 Video. Content-based interactivity is one of the most important novelties offered by MPEG-4. Coding and representing video objects rather than video frames enables content-based applications. Robustness in error-prone environments allows MPEG-4 encoded content to be accessible in a wide range of media, such as mobile networks as well as wired connections. In addition, object-based temporal and spatial scalabilities allow the user to decide where to use sparse resources, which can be the available bandwidth, but also the computing capacity or power consumption. To support some of these functionalities, MPEG-4 should provide the capability to represent arbitrarily shaped video objects. Each object can be encoded with different parameters and at different qualities. The shape of a video object can be represented in MPEG-4 by a binary or a gray-level (alpha) plane. The texture is coded separately from its shape. To increase robustness to errors, special provisions are taken into account at the bit-stream level to allow fast resynchronization and efficient error recovery. The MPEG-4 Video has been explicitly optimized for three bit rate ranges: below 64 Kb/s, 64 to 384 Kb/s and 384 Kb/s to 4 Mb/s. Both CBR and VBR are supported.

An MPEG-4 Video bit stream provides a hierarchical description of a visual scene as shown in Figure 5.42. Each level of the hierarchy can be accessed in the bit stream by special code values called start codes. The hierarchical levels that describe the scene most directly are Visual Object Sequence (VS), Video Object (VO), Video Object Layer (VOL), Group of Video Object Planes (GOV) and Video Object Plane (VOP).

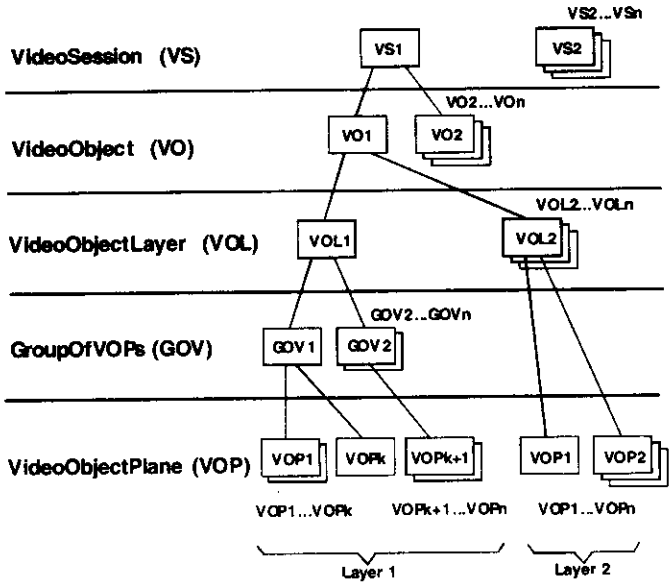

Figure 5.42 MPEG-4 video bit stream logical structure.
©2000 ISO/IEC.

By VS, we mean the complete MPEG-4 scene, which may contain any 2D or 3D natural or synthetic objects and their enhancement layers.

VO corresponds to a particular 2D object in the scene. In the most simple case, this can be a rectangular frame, or it can be an arbitrarily shaped object corresponding to an object or background of the scene.

VOL provides support for scalable coding. Each video object can be encoded in scalable (multilayer) or nonscalable form (single layer), depending on the application, represented by the VOL. Going from the coarse to fine resolution, a VO can be encoded using spatial or temporal scalability. There are two types of VOs: the VOL that provides full MPEG-4 functionality and a reduced functionality VOL, which is the VOL with short headers [5.66]. Each VO is sampled in time, and each time sample of a VO is a VOP. VOPs can be grouped together to form a GOV.

The GOV groups together VOPs. GOVs can provide points in the bit stream where VOPs are encoded independently from each other. Thus, they can provide random access points into the bit stream.

VOP is a time sample of a VO [5.67]. VOPs can be encoded independently of each other or dependent on each other by using motion compensation. In the most common way, the VOP contains the encoded video data of a time sample of a VO. In that case, it contains motion parameters, shape information and texture data. These are encoded using macroblocks [5.68]. It can also be used to code a sprite. A sprite is a VO that is usually larger than the display video presented over time. It is used to represent large, more or less static areas, such as backgrounds. Sprites are encoded using macroblocks. Each macroblock contains four luminance blocks and two chrominance blocks. Each block contains 8x8 pixels and is encoded using the DCT [5.69]. A macroblock carries the shape information, motion information and texture information. Figure 5.43 illustrates the general block diagram of MPEG-4 coding and decoding based on the VOs. Each video object is coded separately. For reasons of efficiency and backward compatibility, video objects are coded through their corresponding VOPs in a hybrid coding scheme. MPEG-4 coding for natural video will also provide tools that enable a number of other functionalities such as object scalability, spatial and temporal scalabilities, sprite overlays, error resilience, and so forth. MPEG-4 Video is capable of coding conventional rectangular video as well as arbitrarily shaped 2D objects in a video scene. It will be able to code video ranging from very low spatial and temporal resolutions in progressive scanning format up to very high spatial and temporal resolutions for professional studio applications, including interlaced video.

In what follows, we will discuss coding tools (shape coding, motion estimation and compensation, texture coding and multifunctional coding), error resilience, sprite coding and scalability.

Shape-Coding Tools for MPEG-4 Natural Video

Besides the shape information available for the VOP in question, the shape-coding scheme also relies on motion estimation to compress the shape information [5.70]. The shape and location of VOPs may vary from one VOP to the next. The shape may be conveyed either implicitly or explicitly. With implicit shape coding, the irregularly shaped object is placed in front of a colored

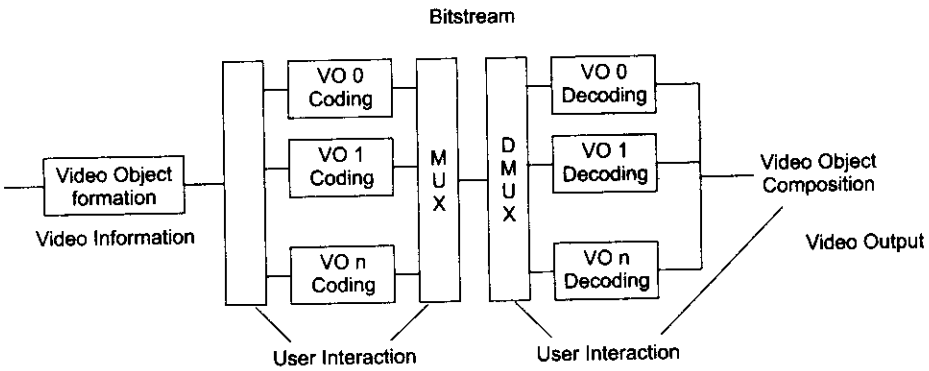

Figure 5.43 General block diagram of MPEG-4 video coding and decoding [5.62]. ©1999 ISO/IEC.

background known to receiver. A rectangular VOP containing both the object and background is coded and transmitted. The decoder retrieves the object by simple chromakeying.

Explicit shape is represented by a rectangular alpha plane that covers the object to be coded. An alpha plane may be binary if only the shape is of interest ("0" for background or "1" for object). It may be also gray level (up to 8 bits per pixel) to indicate various levels of partial transparency for the object. If the alpha plane has a constant gray value inside the object area, that value can be sent separately, and the alpha plane can be coded as a binary alpha plane. One of the most promising algorithms for coding alpha planes is arithmetic coding.

In the MPEG-4 Visual standard, two kinds of shape information are considered as inherent characteristics of a VO. These are referred to as binary and gray shape information. By binary shape information, one means label information that defines which partitions (pixels) of the support of the object belong to the VO at a given time. This kind of information is most commonly represented as a matrix with the same size as that of the bounding box of a VOP. Every element of the matrix can take one of the two possible values, depending on whether the pixel is inside or outside of the VO. Gray-scale shape is generalization of the concept of binary shape providing a possibility to represent transparent objects and to reduce aliasing effects.

Motion Estimation and Compensation

Motion estimation and compensation are commonly used to compress video sequences by exploiting temporal redundancies among frames. The main difference between MPEG-4 and other standards concerning motion compensation is that the block-based techniques used in the other standards have been adopted to the VOP structure in MPEG-4. Three modes for encoding an input VOP are provided by MPEG-4 as shown in Figure 5.44. Namely, Intra VOPs (I-VOPs) are coded without any information from other VOPs. Predicted VOPs and Bidirectional Interpolated VOPs (P- and B-VOPs) are predicted based on I- and/or P-VOPs.

A VOP may be encoded independently of any other VOP. In this case, the encoded VOP is called an I-VOP. A VOP may be predicted using motion compensation based on another previ-

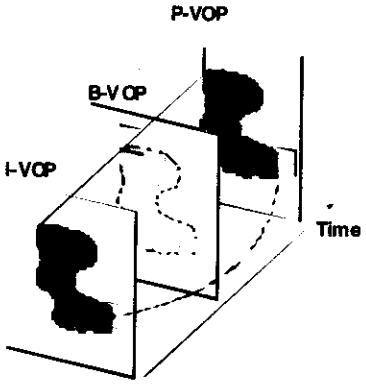

Figure 5.44 The three modes of VOP encoding. ©1999 ISO/IEC.

ously decoded VOP. Such VOPs are called P-VOP. A VOP may be predicted based on past as well as future VOPs. Such VOPs are called B-VOPs.

Motion Estimation (ME) is necessary only for coding P-VOPs and B-VOPs. ME is performed only for macroblocks in the bounding box of the VOP in question. If a macroblock lies entirely within a VOP, ME is performed based on block matching of 16x16 macroblocks as well as 8x8 blocks in advance prediction mode. This results in one motion vector for the entire macroblock and one for each of its blocks.

For P- and B-VOPs, the motion vectors are first differentially coded based on up to three motion vectors of previously transmitted blocks. The exact number depends on the allowed range of the vectors. The maximum range is selected by the encoder and transmitted to the decoder. Variable length coding is then used to encode the motion vectors. Here, for each block of the macroblock, the motion vectors of neighboring blocks are considered. This includes the motion vector of the current block and its four neighbors. Each vector provides an estimate of the pixel value. The actual predicted value is then a weighted average of all these estimates.

Texture-Coding Tools

Coding of texture for arbitrarily shaped regions with a shape described with an alpha map is different than traditional methods. For example, intraframe coding, forward prediction motion compensation and bidirectional motion-compensation are used. This gives rise to the definitions of I-VOPs, P-VOPs and B-VOPs for VOPs that are intracoded, forward predicted or bidirectionally predicted, respectively.

The texture information of a VOP is present in the luminance Y and two chrominance components, C_b, C_r, of the video signal. In the case of an I-VOP, the texture information resides directly in the luminance and chrominance components. In the case of motion-compensated VOPs, the texture information represents the residual error remaining after motion compensation. For encoding the texture information, the standard 8x8 block-based DCT is used. To encode an arbitrarily shaped VOP, an 8x8 grid is superimposed on the VOP. Using this grid, 8x8 blocks that are internal to the VOP are encoded without modifications. Blocks that straddle the VOP are called boundary blocks and are treated differently from internal blocks. The transformed blocks are

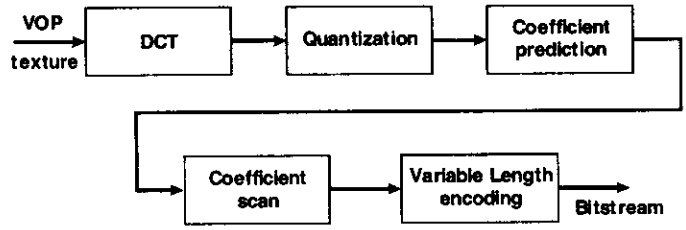

Figure 5.45 Block diagram of VOP texture-coding process. ©1999 ISO/IEC.

quantized, and individual coefficient prediction can be used from neighboring blocks to further reduce the entropy values of the coefficients. This is followed by a scanning of the coefficients to increase the average run length between coded coefficients. Then, the coefficients are encoded by variable-length encoding. This process is illustrated in Figure 5.45.

Internal video texture blocks and padded boundary blocks are encoded using a 2D 8x8 block-based DCT. The DCT coefficients are quantized as a lossy compression step. There are two methods of quantization. The first method uses one of two available quantization matrixes to modify the quantization step size depending on the spatial frequency of the coefficient. The second method uses the same quantization step size for all coefficients. The average energy of the quantized coefficients can be further reduced by using prediction from neighboring blocks. The prediction can be performed from either the block above, the block to the left, or the block above to the left. Candidate blocks for coefficient prediction are illustrated in Figure 5.46. The direction of the prediction is adaptive and is selected based on comparison of horizontal and vertical DC gradients (increase or reduction in its value) of surrounding blocks A, B and C. There are two types of predictions, DC prediction and AC prediction. The DC prediction is performed for the DC coefficient only and is either from the DC coefficient of block A or from the DC coefficient of block C. As for AC prediction, either the coefficients from the first row or the coefficients from the first column of the current block are predicted from the cosited coefficients of the selected candidate blocks.

Before the coefficients are run-length coded, a scanning process is applied to transform the 2D data into a 1D string. Three different scanning methods are possible: zig-zag scan, alternate-horizontal scan and alternate-vertical scan. In zig-zag scan, the coefficients are read out

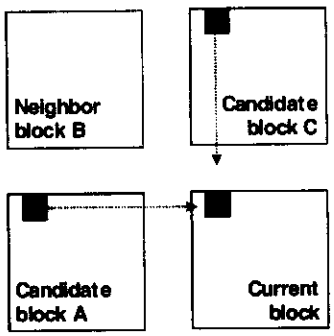

Figure 5.46 Blocks for coefficients prediction. ©1999 ISO/IEC.

diagonally, and, in alternate-horizontal scan, the coefficients are read out with an emphasis on the horizontal direction first. Alternate-vertical scan is similar to the horizontal direction scan, but are applied in the vertical direction. If there is DC prediction in the horizontal direction, the alternate-vertical scan is used, and if DC prediction is performed from the vertical direction, the alternate-horizontal scan is used. Zig-zag scan is used if there is no DC prediction.

The run-length encoding can use two different VLC tables and use the value of the quantizer to determine which VLC table is used [5.65].

Multifunctional Coding

Multifunctional coding refers to features other than coding efficiency. For example, object-based spatial and temporal scalabilities are provided to enable broad-based access across a variety of networks and facilities. This can be useful for Internet and database applications. Spatial scalability with two layers and temporal scalability with two layers are shown in Figure 5.47 and Figure 5.48, respectively.

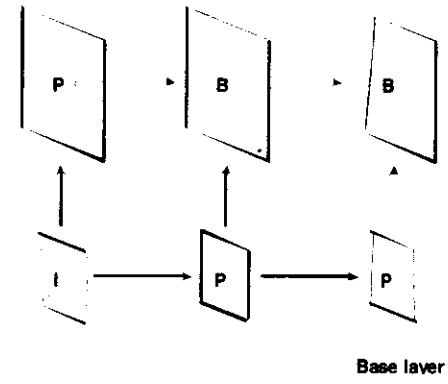

Figure 5.47 Spatial scalability with two layers. ©1999 ISO/IEC.

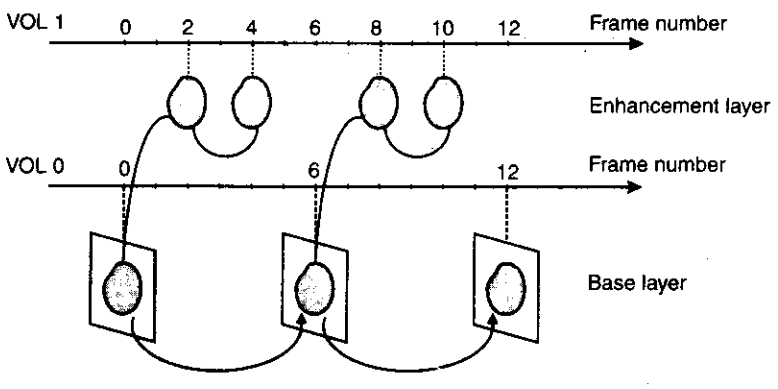

Figure 5.48 Temporal scalability with two layers [5.62]. ©1999 ISO/IEC.

For mobile multimedia applications, spatial and temporal scalabilities are useful for channel bandwidth scaling for robust delivery.

Multifunctional coding also addresses multiview and stereoscopic applications, as well as representations that enable simultaneous coding and tracking of objects for surveillance and other applications. Besides the aforementioned applications, a number of tools are being developed for segmentation of a video scene into objects and for suppressing noise.

Sprite Coding

A sprite consists of those regions of a VO that are present in the scene throughout the video segment. As an example, a background sprite consists of all pixels belonging to the background of a camera-panning sequence. Sprites have been included in MPEG-4 mainly because they provide high efficiency in such cases. For any given instant of time, the background VOP can be extracted.

The texture information for a sprite is represented by one luminance component and two chrominance components. The three components are processed separately. The methods used for processing the chrominance components are the same as those used for the luminance components, after appropriate scaling. Shape and texture information for a sprite are encoded as for an I-VOP. Static sprites are generated before the encoding process begins using the original VOPs. The decoder receives each static sprite before the rest of the video segment. The static sprites are encoded in such a way that the reconstructed VOPs can be generated easily by warping the quantized sprite with the appropriate parameters.

Several possibilities are envisaged for the transmission of sprites. One way is to transmit only a portion of the sprite in the beginning. The transmission portion should be sufficient for reconstructing the first few VOPs. The remainder of the sprite is transmitted, piecewise, as required or as the bandwidth allows. Another method is to transmit the entire sprite in a progressive fashion, starting with a low-quality version and gradually improving its quality by transmitting residual images. A combination of these methods can also be used in practice.

Scalability

Scalability essentially means that the compressed bit stream can be manipulated in a simple manner in order to satisfy constraints on such parameters as bit rates, display resolutions and frame rates or for decoding hardware complexity [5.6]. Generally speaking, this manipulation consists of the extraction of relevant subsets from the compressed bit stream, each of which should represent an efficient compression of the video sequence at the same resolution and distortion [5.71]. The value of scalable compression lies in the fact that the bit stream may be manipulated at any point after the compressed bit stream has been generated. This is significant because, in many important applications, advance knowledge of constraints on resolutions, bit rates or decoding complexities may not be available during compression.

Spatial scalability and temporal scalability are both implemented using multiple VOLs. Consider the case of two VOLs: the base layer and the enhancement layer. For spatial scalability, the enhancement layer improves upon the spatial resolution of a VOP provided by the base layer.

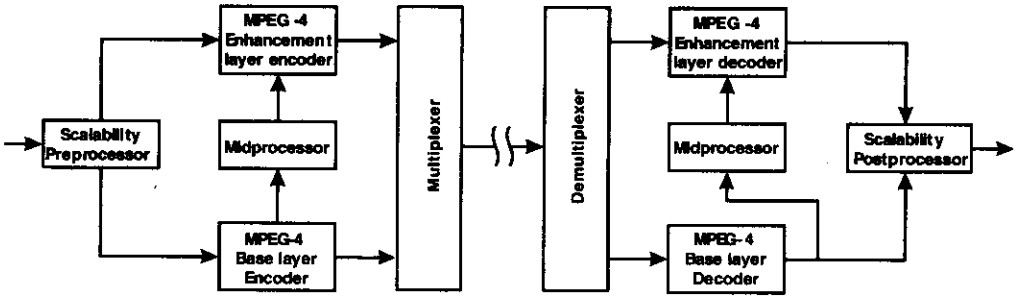

Figure 5.49 Block diagram of MPEG-4 generalized scalability. ©1999 ISO/IEC.

In the case of temporal scalability, the enhancement layer may be decoded if the desired frame rate is higher than that offered by the base layer. Thus, temporal scalability improves the smoothness of motion in the sequence. Object scalability is naturally supported by the VO-based scheme.

MPEG-4 uses a generalized scalability framework to enable spatial and temporal scalabilities. To enable the various scalabilities, this framework allows the inclusion of separate modules. The block diagram of the MPEG-4 generalized scalability framework is presented in Figure 5.49. The specific algorithms implemented in the preprocessor, midprocessor and postprocessor depend upon the type of scalability being enabled. A scalability preprocessor is used to implement the desired scalability. It operates on VOPs. In the case of spatial scalability, the preprocessor down-samples the input VOPs to produce the base layer VOPs that are processed by the VOP encoder. The midprocessor takes the reconstructed base-layer VOPs and up-samples them. The difference between the original VOP and the output of the midprocessor forms the input to the encoder for the enhancement layer. To implement temporal scalability, the preprocessor separates out the frames into two streams. One stream forms the input for the base-layer encoder, and the other is processed by the enhancement layer encoder. The midprocessor is bypassed.

The base-layer VOPs are encoded in the same way as in the nonscalable case discussed in previous sections. VOPs of the enhancement layer are encoded as P-VOPs or B-VOPs.

In scalability coding of a VOP, if the enhancement layer is temporally coincident with an I-VOP in the base layer, it could be treated as a P-VOP. VOPs in the enhancement layer that are coincident with P-VOP in the base layer could be coded as B-VOPs. VOPs in the base layer must be encoded before their corresponding VOPs in the enhancement layer, because the base layer serves as the reference for the enhancement layer.

In temporal scalability, the frame rate of the visual data is enhanced. Enhancement layers carry information to be visualized between the frames of the base layer. The enhancement layer may act in two ways:

- The enhancement layer improves the resolution of only a portion of the base layer.
- The enhancement layer improves the resolution of the entire base layer.

Error Resilience

Error resilience is needed, to some extent, in all transmission media. For an example, due to the rapid growth of mobile communications, wireless networks are typically prone to error and usually operate at relatively low bit rates, for example, less than 64 Kb/s. Both MPEG and ITU-T are working on error-resilience methods, including forward error correction, automatic request for retransmission, scalable coding, slice-based bit-stream partitioning and motion-compensated error correction.

MPEG-4 provides several mechanisms to allow error resilience with different degrees of robustness and complexities [5.39]. These mechanisms are offered by tools providing means for resynchronization, error detection, data recovery and error concealment. There are four error resilience tools in MPEG-4 Visual: resynchronization, data partitioning, header extension code and reversible variable-length codes.

The most frequent way to bring error resilience to a bit stream is resynchronization. It consists of inserting unique markers in the bit stream so that, in the case of an error, the decoder can skip the remaining bits until the next marker and restart decoding from that point on. MPEG-4 allows for insertion of resynchronization markers after an approximately constant number of coded bits or video packets.

The data-partitioning method separates the bits for coding of motion information and those for the texture information. In the event of an error, more efficient error concealment may be applied when, for example, the error occurs on the texture bits only by making use of the decoded motion information.

A header extension code represents the binary codes that allow an optional inclusion of redundant header information, which is vital for correct decoding of video. In this way, the chances of corruption of header information and complete skipping of large portions of the bit stream will be reduced. Reversible Variable Length Coding (RVLC) further reduces the influence of error occurrence on the decoded data. RVLCs are code words that can be decoded in forward as well as backward manners. In the event of an error and skipping of the bit stream until the next resynchronization marker, it is possible to still decode portions of the corrupted bit stream in the reverse order to limit the influence of the error.

Relationship Between Natural and Synthetic Video Coding

The visual objects may have natural or synthetic content, including arbitrary shape VOs, special synthetic objects such as the human face and body and generic 2D/3D objects composed of primitives like rectangles, spheres or indexed face sets, which define an object surface by means of vertexes and surface patches. The synthetic VOs are animated by transforms and special-purpose animation techniques, such as face/body animation and 2D mesh animation. The representation of synthetic VOs in MPEG-4 is based on the prior VRML standard [5.39, 5.56, 5.72, 5.73].

The advent of networked multimedia, for example, across the Internet, has resulted in an increased need for interactive retrieval and composition of natural and synthetic visual content on demand, as well as reuse of archived content at the desktop. This in turn, necessitates new

content-based visual representations and standards that allow compression, manipulation, search, browsing and distribution of synthetic and natural VOs, as well as means to synchronize and composite them at the user terminal [5.74]. An important question in object-based video representation is how to extract or segment video objects. Fortunately, a large portion of material used in movie and television productions today is blue screened (chromakeyed), which is when individual VOs are captured as separate camera shots against a blue background. Composition of natural video clips with synthetic objects with known alpha planes, for example, text overlays, channel logos, animation characters and so on, is commonplace. For conventional video sources, an automatic layering method by motion segmentation was proposed [5.75].

The 2D and 3D mesh models were introduced in the image processing and compression literature for motion compensation. In the computer graphics world, meshes have long been used for 3D shape modeling. A 2D mesh is a planar graph that partitions a 2D image region into polygonal patches. The vertexes of the polygonal patches are referred to as node points. Mostly, the patches are triangles or quadtriangles, leading to triangular or quadrilateral meshes, respectively. Mesh-based motion modeling differs from block-based motion modeling in that the patches overlap neither in the reference frame nor in the current frame. Block-based motion modeling and mesh-based motion modeling are represented in Figure 5.50. Arrows show corresponding patches in the reference image (left) and the current image (right). In both cases, backward motion estimation is shown, by which patches are searched in the reference image that best matches a given patch in the current frame. Polygonal patches in the current frame are deformed by the movements of the node points into polygonal patches in the reference frame. The texture inside each patch in the reference frame is warped [5.76].

In content-based video compression, manipulation and indexing, the shape, motion and texture of each arbitrary-shaped VO need to be modeled and encoded independently. In the MPEG-4 Video verification model, the shape of a VOP is represented by a bitmap, called an alpha plane (binary or gray scale), and the text (color) of the VOP is represented by a texture plane [5.75]. The values of pixels in the texture plane are defined in the corresponding alpha-plane pixel as nonzero.

The 2D mesh representation of VOs enables the following functionalities: VO compression, VO manipulation and content-based video indexing.

VO compression. Mesh modeling may improve compression efficiency in two ways. Namely, the mesh model provides better motion compensation, that is, the translation-block model, which may result in less blocking artifacts at lower bit rates. Alternatively, we can choose to transmit texture maps only at selected key frames and to animate these texture maps for the intermediate frames without sending any predicted error image.

VO manipulation. By VO manipulation, we mean augmented reality, synthetic-object transformation/animation and spatiotemporal interpolation.

Augmented reality means merging virtual (computer-generated) images with real moving images (video) to create enhanced display information. The computer-generated images must remain in perfect registration with the moving real images, which is why tracking is needed.

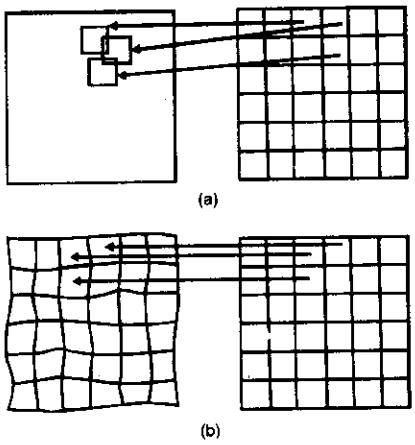

Figure 5.50 (a) 2D block-based motion modeling and (b) 2D quadtrilateral mesh-based motion modeling [5.76]. ©1998 ISO/IEC.

Synthetic-object transformation/animation replaces a natural video object in a video clip with another VO. The replacement VO may be extracted from another natural video clip or may be transfigured from a still-image object using the motion information of the object to be replaced, which is why temporally continuous motion representation is needed.

As for spatiotemporal interpolation, mesh motion modeling provides more robust motion-compensated temporal interpolation compared to block-based motion modeling.

Content-based video indexing. Mesh representation enables animated key snapshots for a moving visual synopsis of an object. Also, it provides accurate object trajectory information that can be used to retrieve VOs with specific motion. Finally, it provides vertex-based object shape representation, which is more efficient than the bitmap representation for shape-based retrieval.

Synthetic Images

MPEG-4 uses VRML as a starting point for its synthetic image specification. MPEG-4 adds a number of additional capabilities. The first addition is a synthetic Face and Body (FAB) animation capability, which is a model-independent definition of artificial FAB animation parameters. With these parameters, one can represent facial expressions, body positions, mouth shapes, and so forth. Planned capabilities include 3D feature point positions and 3D head and body control meshes for animation and texture mapping of face, body and personal characteristics. Also planned is a text-driven mouth animation to be combined with a text-to-speech capability for a complete text-to-talking-head implementation. Another capability being studied is for texture mapping of real image information onto artificial models, such as the FAB model. For this, wavelet-based texture coding is being considered. An advantage of wavelet-based coding is the relative ease of adjusting the resolution of the rendering.

Associated with FAB coding is a triangular mesh modeling capability to handle any type of 2D or 3D synthetic or natural shape. This also facilitates integration of text and graphics onto synthetic and natural images.

Integration of Face Animation with Natural Video

The face object specified by MPEG-4 is a representation of the human face structured in a way that the visual manifestations of speech are intelligible. The facial expressions allow recognition of the speaker's mood and reproduction of a real speaker as faithfully as possible [5.40]. To fulfill these objectives, MPEG-4 specified three types of facial data: Facial Animation Parameters (FAPs), Facial Definition Parameters (FDPs) and FAP Interpolation Table (FIT).

FAPs allow one to animate a 3D facial model available at the receiver. The way by which this model is made available at the receiver is not relevant. FAPs allow the animation of key feature points in the model, independently or in graphs, as well as the reproduction of visemes and expressions.

FDPs allow one to configure the 3D facial model to be used at the receiver, either by adopting a previously available model or by sending a new model.

FIT allows one to define the interpolation rules for the FAPs that have been interpolated at the decoder. The 3D model is then animated using the FAPs sent and the FAPs interpolated according to the FIT.

Although FAPs continuously provide visual information associated with the behavior of the 3D model, FDPs provide model configuration information, which is typically sent only once. For this reason, FAPs are coded as an individual ES—the facial animation stream—while FDPs are fully coded as BIFS nodes and thus are sent in the BIFS stream.

Figure 5.51 shows a general block diagram of a 3D facial animation system, which may fit many applications. For some applications, FAPs and FDPs are extracted from real video input. For many other applications, the analysis phase does not exist. This means that FAP and FDP data are artificially edited to fulfill a certain goal by synthesizing the necessary data. Assuming that FAPs and FDPs follow the standardized bit stream syntax and semantics, the way by which FAPs and FDPs are generalized is completely irrelevant to the receiving terminal.

MPEG-4 specifies a face model in its neutral state, a number of future points on this neutral face as reference points, and a set of FAPs, each corresponding to a particular facial action deforming a face model in its neutral state. Deforming a neutral face model according to some

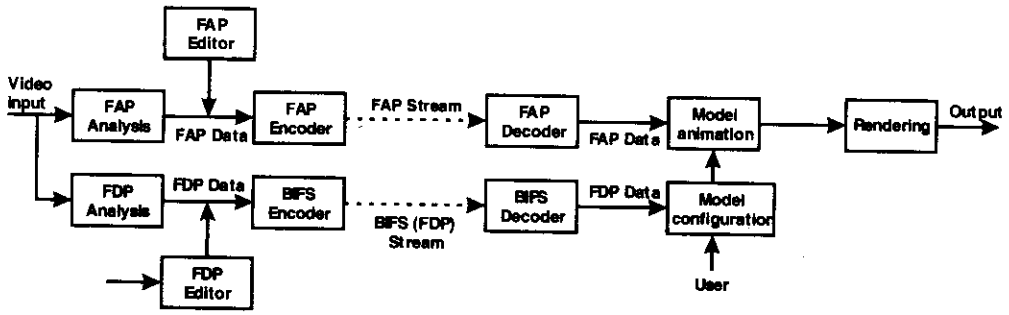

Figure 5.51 General block diagram of a 3D facial animation system [5.39]. ©1998 ISO/IEC.

specified FAP values at each time instant generates a facial animation sequence. The terminal can either use its own animation rules or can download a face model and the associated Facial Animation Tables (FAT) to have a customized animation behavior. Because the FAPs are required to animate faces of different sizes and proportions, the FAP values are defined in Facial Animation Parameters Units (FAPUs). The FAPU is computed from spatial distances between major facial features on the model in its neutral state [5.77]. The FAPU allows interpolation of the FAPs on any facial model in a consistent way, producing results in terms of expression and speech pronunciation. The fractional units used for various FAPs are chosen to allow enough precision for the corresponding FAP.

FAPs

The FAPs are based on the study of minimal perceptible actions and are closely related to muscle actions [5.78]. FAPs were designed to allow the animation of faces, reproducing movements, expressions, emotions and speech pronunciation. The chosen set of FAPs represents a complete set of basic facial movements, allowing terminals to represent most natural facial actions.

The FAP set includes 68 FAPs: 66 low-level parameters associated with the lips, jaw, eyes, mouth, cheek, nose, and so forth, and 2 high-level parameters [5.40]. Although low-level FAPs are associated with movements of key facial zones, typically referenced by a future point, as well as with rotation of the head and eyeballs, visemes and expressions represent more complex actions, typically associated with a set of FAPs. Although the encoder knows the reference feature point for each low-level FAP, it does not precisely know how the decoder will move the model vertexes around that feature point. This is the FAP interpolation model, which describes the specification of the precise changes in the model vertexes corresponding to each FAP.

Table 5.9 shows an excerpt from the FAP list, including number and name, a short description, the specification of the associated FAPU, whether the FAP is unidirectional or bidirectional, the definition of the movement direction for positive values, the group number, the FDP subgroup number and the default quantization step size. MNS is mouth-nose separation [5.39].

With FAP, it is possible to select among six different expressions: joy, sadness, anger, fear, disgust and surprise. Visemes are the visual analog to phonemes and allow the efficient rendering of visemes for better speech pronunciation as an alternative to hearing them represented using a set of low-level FAPs [5.79]. Zero-valued FAPs correspond to a neutral position at the beginning of a session. All FAPs are expressed as displacement from the positions defined for the neutral face. Thus, it is essential to start from neutral faces that are as similar as possible. According to the specification [5.40], the neutral face is characterized by having all muscles relaxed; eyelids tangent to the iris; pupils as one-third of the iris diameter; lips in contact; mouth closed with the upper teeth touching the lower ones and the tongue flat, horizontal and with its tip touching the boundary between upper and lower teeth. Note that the 68 parameters are categorized into 10 groups related to parts of the face. FAP groups are represented in Table 5.10.

The FAP set contains two high-level parameters, visemes and expressions (FAP group 1). A viseme (FAP1) is a visual correlate to a phoneme. Only 14 static phonemes that are clearly distinguished are included in the standard set. Visemes and related phonemes are given in Table 5.11.

Table 5.9 Excerpt of the FAP Specification Table [5.39].

	FAP Name	FAP Description	Units	Uni- or Bidirectional	Motion	Group	FDP Subgroup Number	Default Quantization Step
1	Viseme	Set of values determinining the mixture of two visemes (e.g. pbm, fv, th)	na	na	na	1	na	1
2	Expression	Set of values determinining the mixture of two facial expressions	na	na	na	1	na	1
3	Open_jaw	Vertical jaw displacement (does not affect mouth opening)	MNS	U	down	2	1	4
4	Lower_t_ midlip	Vertical top middle inner lip displacement	MNS	B	down	2	2	2
5	Raise_b_ midlip	Vertical bottom middle inner lip displacement	MNS	B	up	2	3	2

Table 5.10 FAP groups [5.39].

Group	Number of FAPs
1: Visemes and expression	2
2: Jaw, chin, inner lowerlip, cornerlips and midlip	16
3: Eyeballs, pupils and eyelids	12
4: Eyebrow	8
5: Cheeks	4
6: Tongue	5
7: Head rotation	3

Table 5.10 FAP groups [5.39]. (Continued)

Group	Number of FAPs
8: Outer lip positions	10
9: Nose	4
10: Ears	4

Table 5.11 Visemes and related phonemes [5.39].

Viseme No.	Phonemes	Example
0	none	na
1	p, b, m	put, bed, mill
2	f, v	far, voice
3	T, D	think, that
4	t, d	tip, doll
5	k, g	call, gas
6	tS, dZ, S	chair, join, she
7	s, z	sir, zeal
8	n, l	not, lot
9	R	Red
10	A	Car
11	E	Bed
12	I	Tip
13	Q	Top
14	U	Book

In order to allow for coarticulation of speech and mouth movement, the shape of the mouth of a human is not influenced by the current phoneme, but also the previous and the next phoneme. In MPEG-4, transition from one viseme to the next is defined by blending only two visemes with a weighting factor.

The expression parameter FAP2 defines the six primary facial expressions as shown in Table 5.12. In contrast to visemes, facial expressions are animated by a value defining the excitation of the expression. The facial expression parameter values are defined by textual descriptions. The expression parameter allows an efficient means of animation faces. They are high-level animation parameters. A face model designer creates them for each face model.

One way to achieve redundancy reduction is to send only a subset of active FAPs. This subset is then used to determine the values of other FAPs. Such FAP interpolation exploits the symmetry of a human face or the a priori knowledge of articulation functions. For example, the top-inner lip FAPs can be sent and then used to determine the lip-outer-lip FAPs. The inner-lip FAPs would be mapped to the outer-lip FAPs by interpolation.

Table 5.12 Primary facial expressions for FAP2 [5.79].

No.	Expression Name	Textual Description
1	Joy	The eyebrows are relaxed. The mouth is open, and the mouth corners are pulled back toward the ears.
2	Sadness	The inner eyebrows are bent upward. The eyes are slightly closed. The mouth is relaxed.
3	Anger	The inner eyebrows are pulled downward and together. The eyes are wide open. The lips are pressed against each other or opened to expose the teeth.
4	Fear	The eyebrows are raised and pulled together. The inner eyebrows are bent upward. The eyes are tense and alert.
5	Disgust	The eyebrows and eyelids are relaxed. The upper lip is raised and curled, often asymmetrically.
6	Surprise	The eyebrows are raised. The upper eyelids are wide open, and the lower eyelids are relaxed. The jaw is opened.

Example 5.13 MPEG-4 defines a generic face model in its neutral state (Figure 5.52), with the following properties:

- Gaze is in the direction of z axis.
- All face muscles are relaxed.
- Eyelids are tangent to the iris.

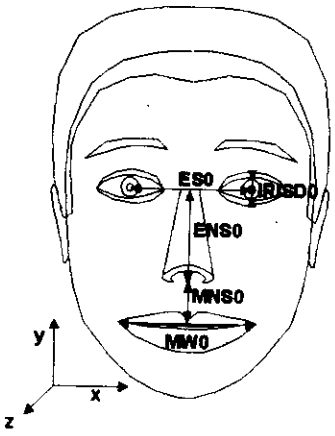

Figure 5.52 A face model in its neutral state and the feature points used to define FAP units [5.39]. ©1998 ISO/IEC.

- The pupil is one-third of the diameter of the iris.
- Lips are in contact, and the line of the lips is horizontal and the same height of lip corners.
- The mouth is closed and the upper teeth touch the lower ones.
- The tongue is flat and horizontal with the lip or tongue touching the boundary between upper and lower teeth.

In order to define face animation parameters for an arbitrary face model, MPEG-4 defines FAP units that serve to scale facial animation parameters for any face model. FAPUs are defined as a fraction of the distances between key facial features. These features, such as eye separation, are defined on a face model that is in the neutral state. The FAPU allows interpolation 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 Table 5.13.

Table 5.13 FAPUs and their definitions [5.39].

Measurement	Description	Measurement Unit
IRISD0	Iris diameter (by definition it is equal to the distance between upper and lower eyelids) in neutral face	IRISD=IRISD0/1024
ES0	Eye separation	ES=ES0/1024
ENS0	Eye-nose separation	ENS=ENS0/1024
MSN0	Mouth-nose separation	MNS=MNS0/1024

Table 5.13 FAPUs and their definitions [5.39]. (Continued)

Measurement	Description	Measurement Unit
MW0	Mouth width	MW=MW0/1024
AU	Angle unit	10E-05 rad

Face Model

Every MPEG-4 terminal that is able to decode FAP streams has to provide an MPEG-4 compliant face model that is animate. Usually, this is a model proprietary to the decoder. The encoder does not know about the look of the face model. Using an FDP node, MPEG-4 allows the encoder to specify completely the face model to animate. This involves defining the static geometry of the face model in its neutral state using a scene graph, defining the surface properties and defining the animation rules using FATs that specify how this model gets deformed by the facial animation parameters. Alternatively, the FDP node can be used to calibrate the proprietary face model of the decoder.

A decoder may choose to specify the location of all or some feature points. Then, the decoder is supported to adapt its own proprietary face model such that the model conforms to the feature point positions. Because MPEG-4 does not specify any algorithm for adapting the surface of the proprietary model to the new feature point locations, we cannot specify the subjective quality of a face model after its adaptation. Face model adaptation also allows for the downloading of texture maps for the face. In order to specify the mapping of the texture map onto the proprietary face model, the encoder sends texture coordinates for each feature point. Each texture coordinate defines the location of one feature point on the texture map. This does not allow for precise texture mapping at important features like eyelids or lips. This process of adapting the feature point locations of a proprietary face model according to encoder specifications is referred to as face model calibration. MPEG-4 does not specify any minimum quality of the adapted face model. Therefore, sometimes, this process is called face model adaptation. A method for face model adaptation has been proposed [5.80, 5.81]. An iterative approach based on radial basis functions for scattered data interpolation is used. For each feature point of the proprietary model, a region of interest is defined. When a feature point moves, it deforms the model within this region of interest. The advantage of face model adaptation over downloading a face model from the encoder to the decoder is that the decoder can adapt its potentially very sophisticated model to the desired shape.

In order to download a face model to the decoder, the encoder specifies the static geometry of the head model with a scene graph using MPEG-4 BIFS. VRML and BIFS describe scenes as a collection of nodes arranged in a scene graph. Three types of nodes are of particular interest for the definition of a static head model. A group node is a container for collecting child objects. It allows for building hierarchical models. For objects to move together as a group, they need to

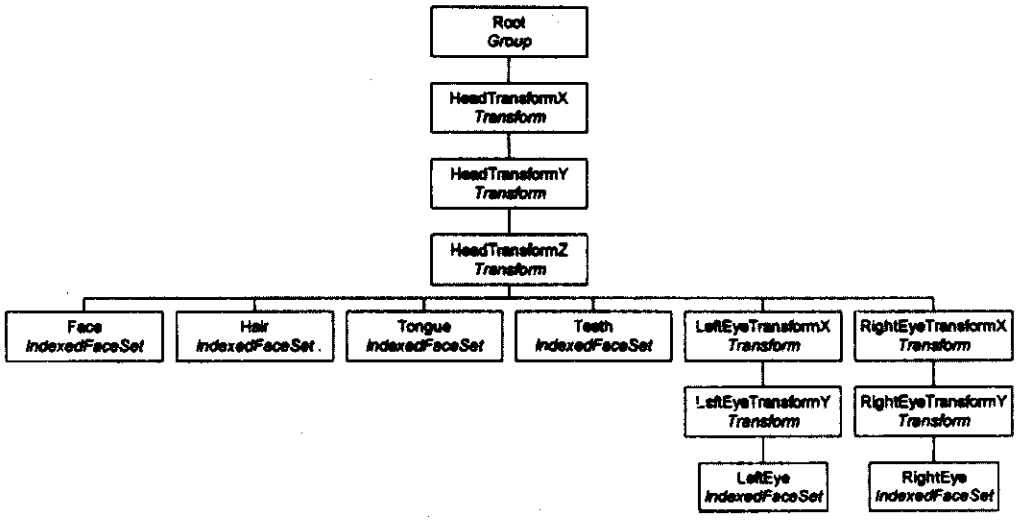

Figure 5.53 Simplified scene graph for a head model. ©1999 ISO/IEC.

be in the same transform group. The transform node defines geometric affine 3D transformations like scaling, rotation and translation that are performed on its children. When transform nodes contain other transforms, their information settings have a cumulative effect. Nested transform nodes can be used to build a transformation hierarchy. An indexed face set node defines the geometry (3D mesh) and surface attributes (color and texture) of a polygonal object. Texture maps are coded with the wavelet coder of the MPEG still-image coder [5.62].

The simplified scene graph for a head model is shown in Figure 5.53. Nested transforms are used to apply rotations about the x,y and z axes, one after another. Embedded into these global head movements are the rotations for the left and right eyes. Separate indexed face sets define the shape and the surface of the face, hair, tongue, teeth, left eye and right eye, thus allowing for separate texture maps. Because the face model is specified with a scene graph, this face model can be easily extended to a head and shoulder model. The surface of the face can be specified using colors or still images to define texture-mapped models. The shape of the face models can be generated using interactive models, scanners or image analysis software [5.82].

Coding of FAPs

MPEG-4 provides two tools for coding FAPs. Coding of quantized and temporally predicted FAPs using an arithmetic coder allows for low-delay FAP coding. Alternatively, DCT coding of a sequence of FAPs introduces a larger delay, but achieves higher coding efficiency. Figure 5.54 shows the block diagram for low-delay encoding of FAPs. The first set of FAP values, FAP_0, in time instant 0 is coded without prediction (intracoding). The value of a FAP at time instant k, FAP_k, is predicted using the previously decoded value FAP_{k-1}. The prediction error e' is quantized using a quantization step size QP multiplied by a quantization parameter FAP_QUANT with 0<FAP_QUANT<31. FAP_QUANT is identical for all FAP values at one time instant k. Using

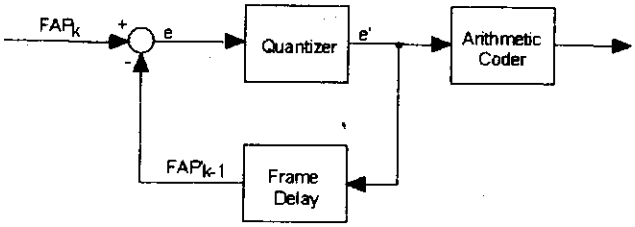

Figure 5.54 Block diagram of the low-delay encoder for FAPs. ©1999 ISO/IEC.

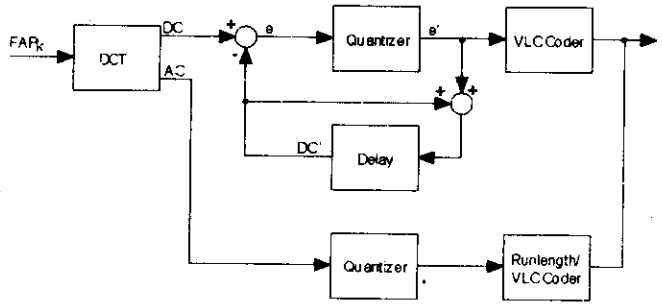

Figure 5.55 Block diagram of the FAP encoder using DCT. ©1999 ISO/IEC.

FAP-dependent quantization step size QP, FAP_QUANT ensures that quantization errors are subjectively evenly distributed between different FAPs. The quantized prediction error e' is arithmetically encoded using a separate adaptive probability model for each FAP. Because the encoding of the current FAP value depends only on one previously coded FAP value, this coding scheme allows for low-delay communications. At the decoder, the received data is arithmetically decoded, dequantized and added to the previously decoded value in order to recover the encoded FAP value. When using FAP_QUANT>15, the subjective quality of the animation deteriorates significantly such that it can be recommended not to increase FAP_QUANT greater than 15 [5.79].

The second tool that is provided for encoding FAPs is the DCT applied to 16 consecutive FAP values. This introduces a significant delay in the coding and decoding processes. Hence, this coding method is mainly useful for applications where animation parameter streams are retrieved from a database. After computing the DCT of 16 consecutive values of one FAP, DC and AC coefficients are coded differently as shown in Figure 5.55.

The DC value is coded predictively using the previous DC coefficient as prediction, and the AC coefficients are directly coded. The AC coefficients and the prediction error of the DC coefficient are uniformly quantized. The quantizer step size can be controlled. The ratio between the quantizer step size of the DC coefficients and the AC coefficients is set to 1:4. The quantized AC coefficients are encoded with one VLC word defining the number of zero coefficients prior to the next nonzero coefficient and one VLC for the amplitude of this nonzero coefficient. The handling of the decoded FAPs with respect to masking and interpolation is not changed. In contrast to the arithmetic coder, the DCT coder is not able to code FAPs with near-lossless quality. At low data rates, the DCT coder requires up to 50% less data rate than the arithmetic coder at