6.5

MPEG-4, H.264/AVC, and MPEG-7: New Standards for the Digital Video Industry

Berna Erol
Ricoh Innovations California Research Center

Adriana Dumitraş
Apple Computer, Cupertino

Faouzi Kossentini
UB Video, British Columbia

Anthony Joch
LSI Logic, Waterloo

Gary Sullivan
Microsoft Corporation, Redmond

1 Introduction

The last two decades have witnessed an increasing number of digital video applications in many areas, including communications, education, medicine and entertainment. Such video applications improve interpersonal communication, promote faster understanding of complex ideas, provide increased access capabilities to information, and allow higher levels of interactivity with the video.

The vast amount of digital data that is associated with video applications and the complex interactions between the different types of video data make the representation, exchange, storage, access, and manipulation of this data a challenging task. To provide interoperability between different video applications and promote further use of video data, there is a need to standardize the representation of, and access to, this data. There had already been significant work in the fields of efficient representation of video by means of compression, storage, and transmission [1–4]. However, the new generation of highly interactive video applications requires that the users be able to access and manipulate video data very efficiently and interactively. Realizing that a single standard may not be able to address all of these requirements, Working Group 11 of the ISO/IEC JTC1/SC29 technical committee, also known as the Moving Picture Experts Group (MPEG), recently developed two new standards: MPEG-4, which standardizes a content-based coded representation of multimedia data, and MPEG-7, which standardizes a multimedia content description interface.

MPEG-4, and in particular Part 10 (better known as H.264/AVC [5]), which is the result of a joint project between MPEG and the Video Coding Experts Group (VCEG) of the ITU-T, offers high compression and error-resilience performance levels, making the representation of video much more efficient and robust. MPEG-4 also enables content-based access and provides functionalities such as scalability and hybrid coding of natural and synthetic video [6]. On the other hand, MPEG-7 enables effective and efficient content-based access and manipulation of video data and provides functionalities that are complementary to those of the MPEG-4 standard. In what follows, we provide a technical description of the two video parts of the MPEG-4 standard, followed by

a discussion of some MPEG-4 video applications. Next, we present a technical description as well as a discussion of some applications of the video part of the MPEG-7 standard. We conclude this chapter with the authors' assessment of the future impact of such standards on the fast-evolving digital video industry.

2 Terminology

In this section, we describe some of the terminology used in this chapter. More information on the terminology can be also found in Chapters 6.1 and 6.4. Throughout this chapter, we will use the term *picture* for a frame or a field that is coded or processed as a distinct unit for compression or other processing. A picture consists of a number of arrays of samples. Typically, there are three such arrays, one for each of the axes of the color space used in the picture. For example, the color axes may be Y (also called *luma* and representing monochrome brightness), Cb (the color deviation from monochrome towards pure Blue), and Cr (the color deviation from monochrome towards pure Red).

Each *sample* consists of the intensity of light along the dimension of one color space axis in one elementally small area of a sampling grid. When using a Y, Cb, Cr color space, the resolution (the size of the array in width and height) of the Cb and Cr arrays is typically lower (i.e., the array width and height are smaller) than for the corresponding Y array for the same picture. Note that the term *sample* is used here for precision, rather than the widely used term *pixel*. The former is simply an integer value associated with a spatial location and a single axis of the color space, whereas the latter is a point in the image that has a color associated with it (the color typically being specified in terms of three or more color space axes). Therefore, a discrete cosine transform (DCT), for instance, cannot be computed using pixels, but can be computed using luma samples or chroma samples. The term *pixel* is particularly problematic when the resolution of the picture is not the same for all of the axes of the color space, resulting in a lack of a clearly-specified color at the location of each sample.

A frame contains two fields. A *field* consists of half of the samples in a frame, based on the lines (i.e., the rows of the arrays) in which the samples are found in each array. A top field consists of the even-numbered lines of samples in a frame (counting line 0 as the top line of the frame) and a bottom field consists of the odd-numbered lines of samples. Typically, fields are only discussed in the context of interlaced video, in which the two fields of each frame are sampled at different instants in time.

3 MPEG-4: Technical Description

MPEG-4 version 1 became an International Standard in 1999. The second version of the standard, which also includes the entire technical content of the first version of the standard, became an International Standard in 2000. After these two versions, more parts were developed, and new tools and profiles were introduced. But perhaps most important was the development of Part 10, better known as H.264 or Advanced Video Coding (AVC), which substantially improves MPEG-4's video compression efficiency.

The MPEG-4 standard addresses system issues such as the multiplexing and composition of audiovisual data in the Systems part [7], the decoding of the visual data in Part 2 [8] and Part 10 [9], and the decoding of the audio data in the Audio part [10]. In this chapter, we will focus on the visual parts of the standard, i.e., MPEG-4 Part 2 (MPEG-4 Video) and Part 10 (H.264/AVC).

3.1 MPEG-4 Part 2

MPEG-4 Part 2, officially known as ISO/IEC 14496-2 [8], standardizes an efficient object-based representation of video. Such representation is achieved by defining visual objects and encoding them into separate bitstream segments [6,7]. While MPEG-4 Part 2 defines only the bitstream syntax and the decoding process, the precise definitions of some compliant encoding algorithms are presented in two verification models: one for natural video coding [11], and the other for synthetic and natural hybrid video coding (SNHC) [12].

MPEG-4 Part 2 allows four different types of coding tools: *Video object coding* for the coding of natural and/or synthetic generated, rectangular or arbitrarily shaped video objects, *mesh object coding* for the coding of visual objects represented with mesh structures, *model-based coding* for the coding of a synthetic representation and animation of the human face and body, and *still texture coding* for the wavelet coding of still textures.

In the following sections, we first describe the object-based representation and each of the MPEG-4 Part 2 coding tools. Next, we discuss the scalability and the error resilience tools, followed by a presentation of the MPEG-4 Part 2 profiles.

3.1.1 Object-based Representation

The object-based representation in MPEG-4 Part 2 is based on the concept of the audiovisual object (AVO). An AVO consists of a visual object component, an audio object component, or a combination of these components. The characteristics of the audio and visual components of the individual AVOs can vary, such that the audio component can be (a) synthetic or natural or (b) mono, stereo, or multichannel (e.g., surround sound), and the visual component can be natural or synthetic. Some examples of AVOs include object-based representations of a person recorded by a video camera, a sound clip recorded with a microphone, and a three-dimensional (3D) image with text overlay.

MPEG-4 supports the composition of a set of audiovisual objects into a scene, also referred to as an *audiovisual scene.*

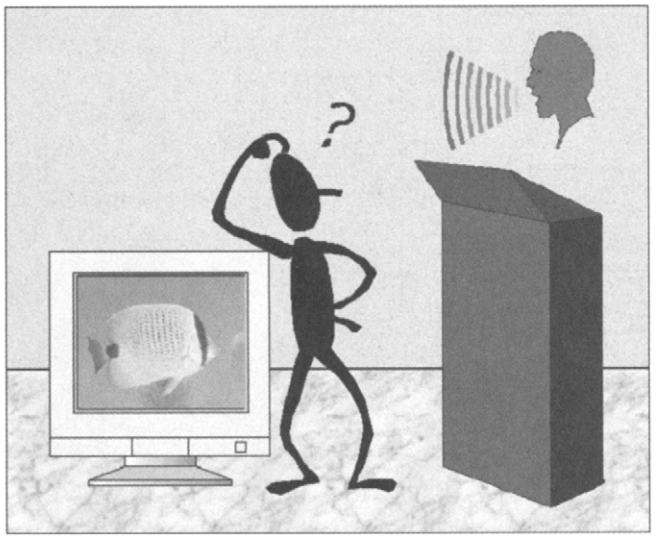


FIGURE 1 An audiovisual scene.

To allow interactivity with individual AVOs within a scene, it is essential to transmit the information that describes each AVO's spatial and temporal coordinates. This information is referred to as the *scene description information* and is transmitted as a separate stream and multiplexed with AVO elementary bitstreams so that the scene can be composed at the user's end. This functionality makes it possible to change the composition of AVOs without having to change the content of AVOs.

An example of an audiovisual scene, which is composed of natural and synthetic audio and visual objects, is presented in Fig. 1. AVOs can be organized in a hierarchic fashion. Elementary AVOs, such as the blue head and the associated voice, can be combined together to form a compound AVO, (i.e., a talking head). It is possible to change the position of the AVOs, delete them or make them visible, or manipulate them in a number of ways depending on their characteristics. For example, a visual object can be zoomed and rotated by the user. Even more, the quality, spatial resolution, and temporal resolution of the individual AVOs can be modified. For example, in a mobile video telephony application, the user can request a higher frame rate and/or spatial resolution for the talking person than those of the background objects.

3.1.2 Video Object Coding

A video object (VO) is an arbitrarily shaped video segment that has a semantic meaning. A 2D snapshot of a video object at a particular time instant is called a video object plane (VOP). A VOP is defined by its texture (luma and chroma values) and its shape. MPEG-4 Part 2 allows object-based access to the video objects, as well as temporal instances of the video objects (i.e., VOPs). To enable access to an arbitrarily shaped object, a separation of the object from the background and the other objects must be performed. This process, known as segmentation, is not standardized in MPEG-4. However, automatic and semi-automatic tools [13] and techniques such as chroma keying [14], although not always effective, can be used for video object segmentation.

As illustrated in Fig. 2, a basic VOP encoder consists mainly of two blocks: a DCT-based motion-compensated hybrid video texture coder, and a shape coder. Similar to[1] MPEG-1 and MPEG-2, MPEG-4 supports intra coded (I-), temporally predicted (P-), and bidirectionally predicted (B-) VOPs, all of which are illustrated in Fig. 3. Except for I-VOPs, motion estimation and compensation are applied. Next, the difference between the motion compensated data and the original data is DCT transformed, quantized, and then variable length coded (VLC). Motion information is also encoded using VLCs. Since the shape of a VOP may not change significantly between consecutive VOPs, predictive coding is used to reduce temporal redundancies. Thus, motion estimation and compensation are also applied to the shape of the VOP. Finally, the motion, texture, and shape information is multiplexed with the headers to form the coded VOP bitstream. At the decoder end, the VOP is reconstructed by combining motion, texture, and shape data decoded from the bitstream.

3.1.2.1 Motion Vector Coding. Motion vectors (MVs) are predicted using a spatial neighborhood of three MVs and the resulting prediction error is variable length coded. Motion vectors are transmitted only for P-VOPs and B-VOPs. MPEG-4 Part 2 uses a variety of motion compensation techniques, such as the use of unrestricted MVs (motion vectors that are allowed to point outside the coded area of a reference VOP), and the use of four MVs per macroblock. In addition, version 2 of MPEG-4 Part 2 supports global motion compensation and quarter–sample motion vector accuracy.

3.1.2.2 Texture Coding. Intra blocks, as well as motion compensation prediction error blocks, are texture coded. Similar to MPEG-1, MPEG-2 (described in Chapter 6.4) and H.263, DCT based coding is employed to reduce spatial redundancies. That is, each VOP is divided into macroblocks as illustrated in Fig. 4. Each macroblock consists of a 16×16 array of luma samples and two corresponding 8×8 arrays of chroma samples. These arrays are partitioned into 8×8 blocks for DCT processing. DCT coding is applied to the four 8×8 luma and two 8×8 chroma blocks of each macroblock. If a macroblock lies on the boundary of an arbitrarily shaped VOP, then the samples that are outside the VOP are padded before DCT coding. As an alternative, a shape-adaptive DCT

[1]When the VOP is a rectangularly shaped video frame, MPEG-4 video coding becomes quite similar to that specified in MPEG-1, MPEG-2 [1, 2] and H.263 [3]. In fact, an MPEG-4–compliant decoder must be able to decode all the bitstreams generated by H.263 baseline compliant encoders.

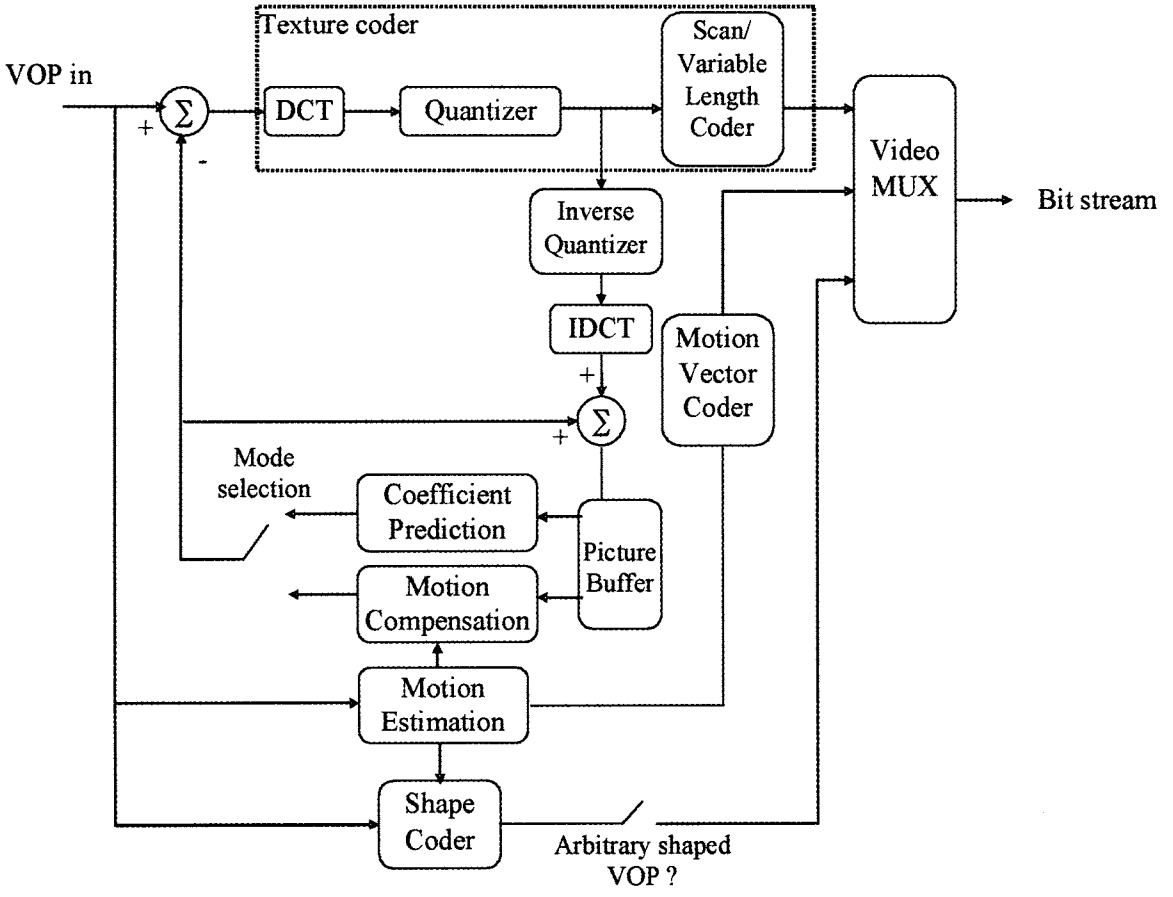


FIGURE 2 A basic block diagram of an MPEG-4 Part 2 video encoder.

coder can be used for coding boundary macroblocks of intra VOPs. This generally results in higher compression performance, at the expense of an increased implementation complexity. Macroblocks that are completely inside the VOP are DCT transformed as in MPEG-1/2 and H.263. DCT transformation of the blocks is followed by quantization, zig-zag coefficient scanning, and variable length coding. Adaptive DC/AC prediction methods and alternate scan techniques can be employed for efficient coding of the DCT coefficients of intrablocks.

3.1.2.3 Shape Coding. Besides H.263+ [4], which provides some limited shape coding support via its chroma-keying coding technique, MPEG-4 Parts 2 and 10 are the only other video coding standards that support shape coding. As opposed

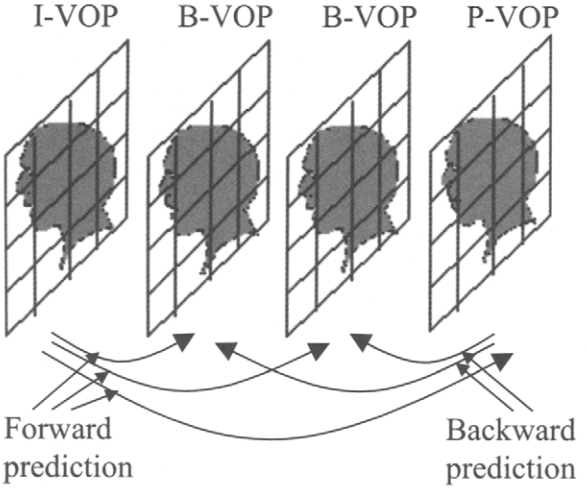


FIGURE 3 Prediction types for a video object plane (VOP).

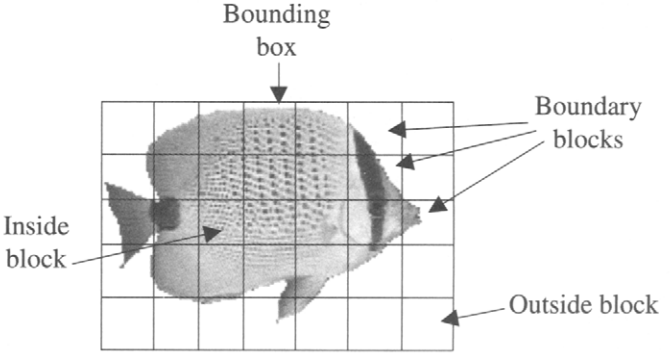


FIGURE 4 A video object plane enclosed in a rectangular bounding box and divided into macroblocks.

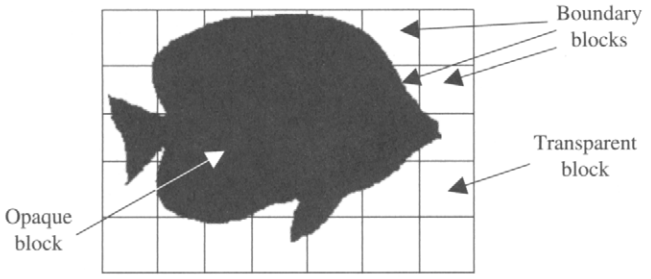


FIGURE 5 Binary alpha plane.

to H.263+, MPEG-4 Part 2 adopted a bitmap-based shape coding method because of its ability to clearly distinguish between shape and texture information while retaining high compression performance and reasonable complexity. In bitmap-based shape coding, the shape and transparency of a VOP are defined by their binary or grayscale (respectively) alpha planes. A binary alpha plane indicates whether or not a sample belongs to a VOP. A grayscale alpha plane indicates the transparency of each sample within a VOP. Transparency of samples can take values from 0 (transparent) to 255 (opaque). If all of the samples in a VOP block are indicated to be opaque or transparent, then no additional transparency information is transmitted for that block.

MPEG-4 Part 2 provides tools for both lossless and lossy coding of binary and gray-scale alpha planes. Furthermore, both intra and inter shape coding are supported. *Binary alpha planes* are divided into 16×16 blocks as illustrated in Fig. 5. The blocks that are inside the VOP are signaled as opaque blocks and the blocks that are outside the VOP are signaled as transparent blocks. The samples in boundary blocks (i.e., blocks that contain both samples inside and outside the VOP) are scanned in a raster scan order and coded using context–based arithmetic coding. *Grayscale alpha planes*, which represent transparency information, are divided into 16×16 blocks and coded the same way as the texture in the luma blocks.

In *intra shape coding* using binary alpha planes, a context is computed for each sample using ten neighboring samples (shown in Fig. 6A) and the equation $C = \sum_k c_k 2^k$, where k is the sample index, c_k is "0" for transparent samples and "1" for opaque samples. If the context samples fall outside the current block, then samples from neighboring blocks are used to build the context. The computed context is then used to access the table of probabilities. The selected probability is used to determine the appropriate code space for arithmetic coding. For each boundary block, the arithmetic encoding process is also applied to the transposed version of the block. The representation that results in the least coding bits can be selected by the encoder to be conveyed in the bitstream.

In *inter shape coding* using binary alpha planes, the shape of the current block is first predicted from the shape of the temporally previous VOP by performing motion estimation and compensation using integer sample accuracy. The shape motion vector is then coded predictively. Next, the difference between the current and the predicted shape block is arithmetically coded. The context for an inter coded shape block is computed using a template of nine samples from both the current and temporally previous VOP shape blocks, as shown in Fig. 6(b).

In both intra and inter shape coding, lossy coding of the binary shape is achieved by either not transmitting the difference between the current and the predicted shape block or subsampling the binary alpha plane by a factor of two or four prior to arithmetic encoding. To reduce the blocky appearance of the decoded shape caused by lossy coding, an upsampling filter is employed during the reconstruction.

3.1.2.4 Sprite Coding. In MPEG-4 Part 2, sprite coding is used for representation of video objects that are static throughout a video scene or are modified such that their changes can be approximated by warping the original object planes [8,15]. Sprites may typically be used for transmitting the background in video sequences. As shown in the example of Fig. 7, a sprite may consist of a panoramic image of the background, including the samples that are occluded by other video objects. Such a representation can increase coding efficiency, since the background image is coded only once in the beginning of the video segment, and the camera motion, such as panning and zooming, can be represented by only a few global motion parameters.

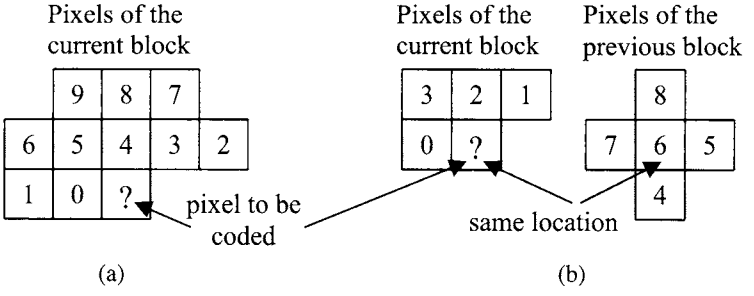


FIGURE 6 Template samples that form the context of an arithmetic coder for (a) intra and (b) inter coded shape blocks.

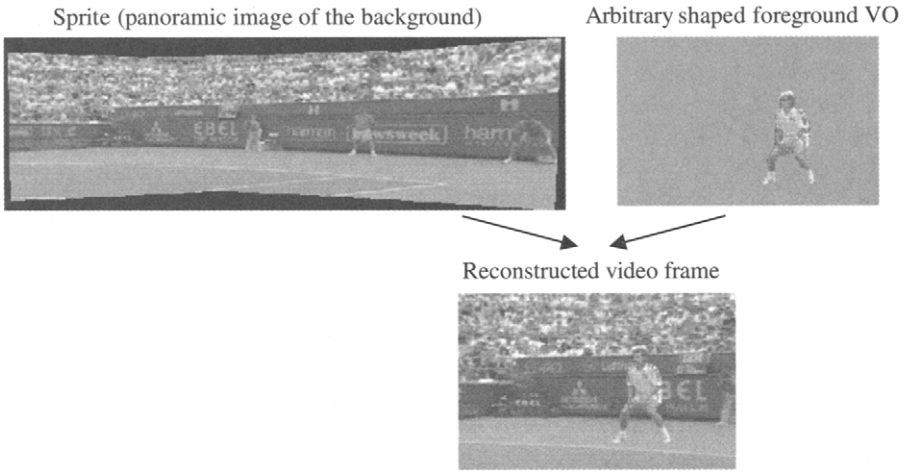


FIGURE 7 Sprite coding of a video sequence (courtesy of Dr. Thomas Sikora, Technical University of Berlin, [6]). VO, video object. (See color insert.)

3.1.3 Mesh Object Coding

A mesh is a tessellation (partitioning) of an image into polygonal patches. Mesh representations have been successfully used in computer graphics for efficient modeling and rendering of 3D objects. In order to benefit from functionalities provided by such representations, MPEG-4 Part 2 supports two-dimensional (2D) and 3D mesh representations of natural and synthetic visual objects, and still texture objects, with triangular patches [8, 16]. The vertices of the triangular mesh elements are called node points, and they can be used to track the motion of a video object, as depicted in Fig. 8. Motion compensation is performed by spatially piecewise warping of the texture maps that correspond to the triangular patches. Mesh modeling can efficiently represent continuous motion, resulting in less blocking artifacts at low bit rates as compared to the block-based modeling. It also enables object-based retrieval of video objects by providing accurate object trajectory information and syntax for vertex-based object shape representation.

3.1.4 Model-based Coding

Model-based representation enables very-low bit rate video coding applications by providing the syntax for the transmission of the parameters that describe the behavior of a human being, rather than transmission of the video frames. MPEG-4 Part 2 supports the coding of two types of models: a *face object* model, which is a synthetic representation of the human face with 3D polygon meshes that can be animated to have visual manifestations of speech and facial expressions, and a *body object* model, which is a virtual human body model represented with 3D polygon meshes that can be rendered to simulate body movements [8, 17, 18].

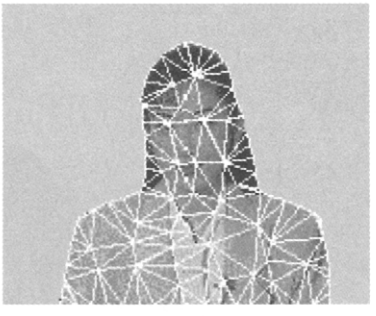


FIGURE 8 Mesh representation of a video object with triangular patches (courtesy of Dr. Murat Tekalp, University of Rochester, [6]).

3.1.4.1 Face Animation. It is required that every MPEG-4 Part 2 decoder that supports face object decoding have a default face model, which can be replaced by downloading a new face model. Either model can be customized to have a different visual appearance by transmitting facial definition parameters (FDPs). FDPs can determine the shape (i.e., head geometry) and texture of the face model.

A face object consists of a collection of nodes, also called feature points, which are used to animate synthetic faces. The animation is controlled by face animation parameters (FAPs) that manipulate the displacements of feature points and angles of face features and expressions. The standard defines a set of 68 low-level animations, such as head and eye rotations, as well as motion of a total of 82 feature points for the jaw, lips, eye, eyebrow, cheek, tongue, hair, teeth, nose, and ear. These feature points are shown in Fig. 9. High level expressions, such as joy, sadness, fear, surprise, and mouth movements are defined by sets of low-level expressions. For example, the joy expression is defined by relaxed eyebrows, and an open mouth with the mouth corners pulled back toward ears. Fig. 10 illustrates several video scenes that are constructed using FAPs.

The FAPs are coded by applying quantization followed by arithmetic coding. The quantization is performed by taking into consideration the limited movements of the facial features. Alternatively, DCT coding can be applied to a vector of 16 temporal instances of the FAP. This solution

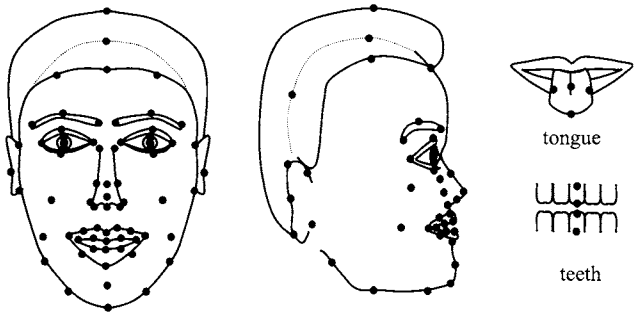


FIGURE 9 Feature points used for animation.

improves compression efficiency at the expense of a higher delay.

3.1.4.2 Body Animation. Body animation was standardized in version 2 of MPEG-4 Part 2. Similar to the case of a face object, two sets of parameters are defined for a body object: body definition parameters (BDPs), which define the body through its dimensions, surface and texture, and body animation parameters (BAPs), which define the posture and animation of a given body model.

3.1.5 Still Texture Coding

The block diagram of an MPEG-4 Part 2 still texture coder is shown in Fig. 11. As illustrated in this figure, the texture is first decomposed using a 2D separable wavelet transform, employing a Daubechies biorthogonal filter bank [8]. The discrete wavelet transform is performed using either integer or floating point operations. For coding of an arbitrarily shaped texture, a shape-adaptive wavelet transform can be used.

The DPCM coding method is applied to the coefficient values of the lowest frequency subband. A multi-scale zero-tree coding method [20] is applied to the coefficients of the remaining subbands. Zero-tree modeling is used for encoding the location of non-zero wavelet coefficients by taking advantage of the fact that, if a wavelet coefficient is quantized to zero, then all wavelet coefficients with the same orientation and the same spatial location at finer wavelet scales are also likely to be quantized to zero. Two different zero-tree scanning methods are used to achieve spatial and quality (signal-to-noise ratio [SNR]) scalability. After DPCM coding of the coefficients of the lowest frequency subband, and zero-tree scanning of the remaining subbands, the resulting data is coded using an adaptive arithmetic coder.

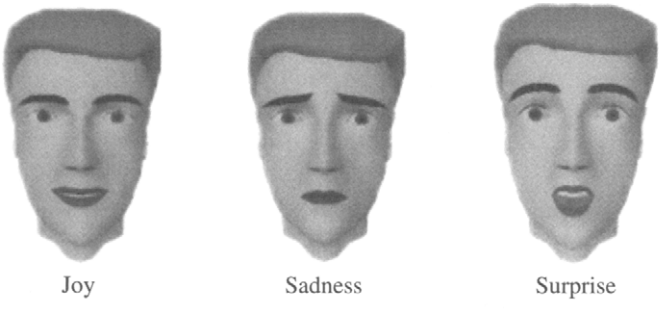


FIGURE 10 Examples of face expressions coded with facial animation parameters (courtesy of Dr. Joern Ostermann, University of Hannover, [19]).

3.1.6 Scalability

In addition to the video coding tools discussed so far, MPEG-4 Part 2 provides scalability tools that allow organization of the bitstream into base and enhancement layers. The enhancement layers are transmitted and decoded depending on the bit rate, display resolution, network throughput, and decoder complexity constraints. Temporal, spatial, quality, complexity, and object-based scalabilities are supported in MPEG-4 Part 2. The first three types of scalability were discussed in Chapter 6.3. Complexity scalability is the scaling of the processing tasks in such a way that the reconstruction quality of the bitstream is adaptable to the processing power of the decoder. Object-based scalability allows the addition or removal of video objects, as well as the prioritization of the objects within a scene. Using this functionality, it is possible to represent the objects of interest with higher spatial and/or temporal resolution, while allocating less bandwidth and computational resources to the objects that are less important. All of these forms of scalability allow prioritized transmission of data, thereby also improving the bitstream's error resilience.

A special case of MPEG-4 Part 2 scalability support, which allows encoding of a base layer and up to eleven enhancement layers, is known as fine granularity scalability (FGS). In FGS coding, the DCT coefficients in the enhancement layers are coded using bit-plane coding instead of the traditional run-level coding [21]. As a result, the FGS coded streams allow for smoother transitions between different quality levels, thereby adapting better to varying network conditions [6, 17, 21].

3.1.7 Error Resilience

To ensure robust operation over error-prone channels, MPEG-4 Part 2 offers a suite of error-resilience tools that can be divided into three groups: resynchronization, data partitioning, and data recovery [8, 22]. *Resynchronization* is enabled by the MPEG-4 Part 2 syntax, which supports a video packet structure that contains resynchronization markers and information such as macroblock number and quantizer in the header. All of these are necessary to restart the decoding operation in case an error is encountered. *Data partitioning* allows the separation between the motion and texture data, along with additional resynchronization markers in the bitstream to improve the ability to localize the errors. This technique provides enhanced concealment capabilities. For example, if texture information is lost, motion information

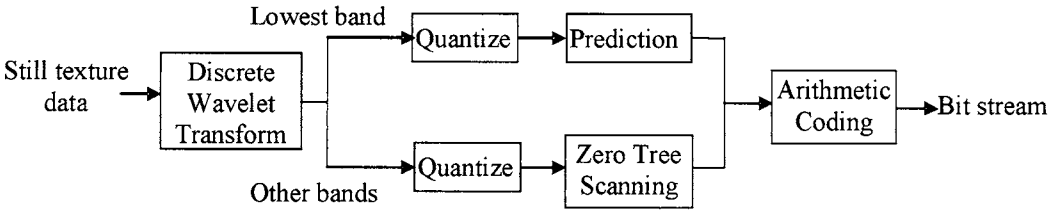


FIGURE 11 Block diagram of the still texture coder.

can be used to conceal the errors. *Data recovery* is supported in MPEG-4 Part 2 by reversible variable-length codes for DCT coefficients and a technique known as new prediction (NEWPRED). Reversible variable length codes for DCT coefficients can be decoded in forward and backward directions. Thus, if part of a bitstream cannot be decoded in the forward direction due to errors, some of the coefficient values can be recovered by decoding the coefficient part of the bitstream in the backward direction. NEWPRED, which is a method intended for real-time encoding applications, makes use of an upstream channel from decoder to encoder, where the encoder dynamically replaces the reference pictures according to the error conditions and feedback received from the decoder [6].

3.1.8 MPEG-4 Part 2 Profiles

Since the MPEG-4 Part 2 syntax is designed to be generic, and includes many tools to enable a variety of video applications, the implementation of an MPEG-4 Part 2 decoder that supports the full syntax is often impractical. Therefore, the standard defines a number of subsets of the syntax, referred to as *profiles*, each of which targets a specific group of applications. For instance, the *simple profile* targets low-complexity and low-delay applications such as mobile video communications, whereas the *main profile* is intended primarily for interactive broadcast and digital video device (DVD) applications. A complete list of the MPEG-4 Part 2 version 1 and version 2 profiles is given in Table 1. The subsequent versions added more profiles to the ones defined in version 1 and version 2, increasing the total number of profiles in MPEG-4 Part 2 to approximately 20. Examples of new profiles include the *advanced simple profile* targeting more efficient coding of ordinary rectangular video, the *simple studio profile* targeting studio editing applications and supporting 4:4:4 and 4:2:2 chroma sampling, and the *fine granularity scalability profile* targeting webcasting and wireless communication applications.

TABLE 1 MPEG-4 Part 2 visual profiles. The index (2) marks profiles that are available in version 2 of MPEG-4 Part 2. The rest of the profiles shown are available in both version 1 and 2 of MPEG-4 Part 2.

Profile Group	Profile Name	Supported Functionalities
Profiles for natural video content	Simple	Error-resilient coding of rectangular video objects
	Simple scalable	Simple profile + frame-based temporal and spatial scalability
	Core	Simple profile + coding of arbitrarily shaped objects
	Main	Core profile + interlaced video coding + transparency coding + sprite coding
	N–bit	Core profile + coding video objects with sample-depths between 4 and 12 bits
	Advanced real-time simple(2)	Improved error resilient coding of rectangular video with low buffering delay
	Core scalable(2)	Core profile + temporal and spatial scalability of arbitrarily shaped objects
	Advanced coding efficiency(2)	Coding of rectangular and arbitrarily shaped objects with improved coding efficiency
Profiles for synthetic and synthetic/natural/hybrid video content	Simple face animation	Basic coding of simple face animation
	Scalable texture	Spatially scalable coding of still texture objects
	Simple basic animated two-dimensional texture	Simple face animation + spatial and quality scalability + mesh-based representation of still texture objects
	Hybrid	Coding of arbitrarily shaped objects + temporal scalability + face object coding + mesh coding of animated still texture objects
	Advanced scalable texture(2)	Scalable coding of arbitrarily shaped texture, shape, wavelet tiling, and error resilience
	Advanced core(2)	Core visual profile + advanced scalable texture visual profile
	Simple face and body animation(2)	Simple face animation profile + body animation

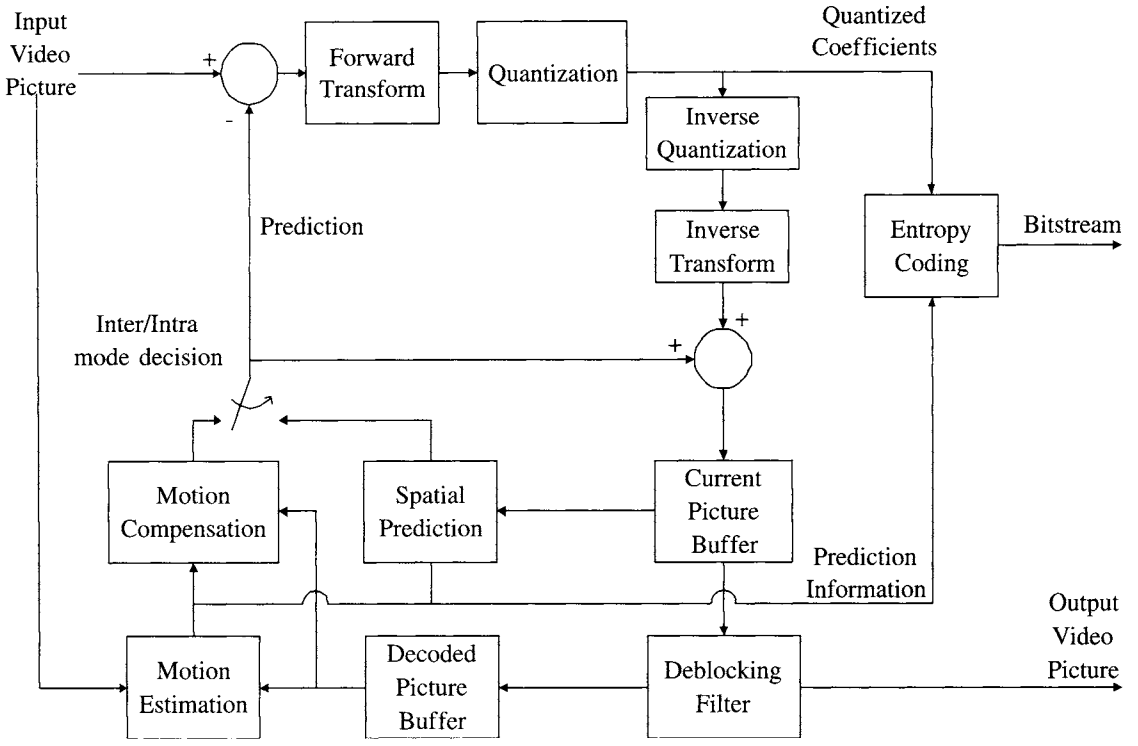


FIGURE 12 Block diagram of an H.264/AVC encoder.

3.2 MPEG-4 Part 10: H.264/AVC

MPEG-4 Part 10: H.264/AVC was developed by a partnership project known as the Joint Video Team (JVT) between the ISO, IEC, and ITU-T, and it became an International Standard in 2003. One of the key goals in the development of H.264/AVC was to address the needs of the many different video applications and delivery networks that would be used to carry the coded video data. To facilitate this, the standard is conceptually divided into a Video Coding Layer (VCL) and a Network Abstraction Layer (NAL). The VCL defines a decoding process that can provide an efficient representation of the video, while the NAL provides appropriate header and system information for each particular network or storage media. The NAL enables network friendliness by mapping VCL data to a variety of transport layers such as RTP/IP, MPEG-2 systems, and H.323. A more detailed description of the NAL concepts and the error resilience properties of H.264/AVC are provided in [23] and [24]. In this chapter, we provide an overview of the H.264/AVC video coding tools that comprise the VCL.

3.2.1 H.264/AVC Video Coding Layer: Technical Overview

Even though H.264/AVC, similar to the preceding standards, defines only the bitstream syntax and video decoding process, here we discuss both the encoding and the decoding process for completeness. H.264/AVC employs a hybrid block-based video compression technique, similar to those defined in earlier video coding standards, which is based on combining inter picture prediction to exploit temporal redundancy and transform-based coding of the prediction errors to exploit the spatial redundancy. A generalized block diagram of an H.264/AVC encoder is provided in Fig. 12. While it is based on the same hybrid coding framework, H.264/AVC features a number of significant components that distinguish it from its predecessors. These features include spatial directional prediction; an advanced motion-compensation model utilizing variable block size prediction, quarter-sample accurate-motion compensation, multiple reference picture prediction, and weighted prediction; an in-loop deblocking filter; a small block-size integer transform and two context-adaptive entropy coding modes.

A number of special features are provided to enable more flexible use or to provide resilience against losses or errors in the video data. The sequence and picture header information are placed into structures known as parameter sets, which can be transmitted in a highly flexible fashion to either help reduce bit rate overhead or add resilience against header data losses. Pictures are composed of slices that can be highly flexible in shape, and each slice of a picture is coded completely independently of the other slices in the same picture to enable enhanced loss/error resilience. Further loss/error resilience is enabled by providing capabilities for data partitioning (DP) of the slice data and allowing redundant picture (RP) slice representations to be sent. Robustness

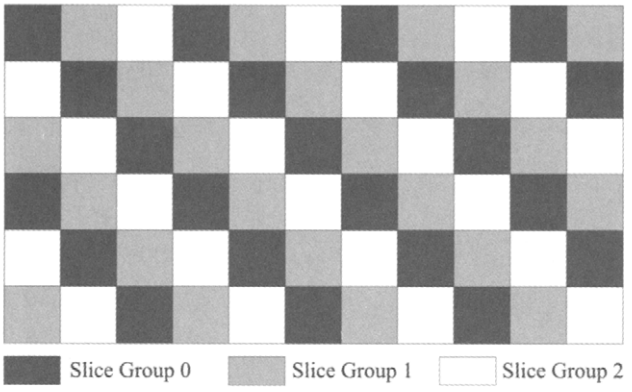


FIGURE 13 Example of slice groups.

against variable network delay is provided by allowing arbitrary slice order (ASO) in the compressed bitstream. A new type of coded slices known as synchronization slices for intra (SI) or inter (SP) coding is supported that enables efficient switching between streams or switching between different parts of the same stream (e.g., to change the server bit rate of pre-encoded streaming video, for trick-mode playback[2] or loss/error robustness).

3.2.1.1 Slices and Slice Groups. As in previous standards, pictures in H.264/AVC are partitioned into macroblocks, which are the fundamental coding unit, each consisting of a 16×16 block of luma samples and two corresponding 8×8 blocks of chroma samples. Also, each picture can be divided into a number of independently decodable slices, where a slice consists of one or more macroblocks. The partitioning of the picture into slices is much more flexible in H.264/AVC than in previous standards. In previous standards, the shape of a slice was often highly constrained and the macroblocks within the same slice were always consecutive in the raster-scan order of the picture or of a rectangle within the picture. However, in H.264/AVC, the allocation of macroblocks into slices is made completely flexible, through the specification of slice groups and macroblock allocation maps in the picture parameter set. Through this feature, informally known as *flexible macroblock ordering* (FMO), a single slice may contain all of the data necessary for decoding a number of macroblocks that are scattered throughout the picture, which can be useful for recovery from errors due to packet loss [23]. An example is given in Fig. 13, which illustrates one method of allocating each macroblock in a picture to one of three slice groups. In this example, the bitstream data for every third macroblock in raster scan order is contained within the same slice group. A macroblock allocation map is specified that categorizes each macroblock of the picture into a distinct slice group. Each slice group is partitioned into one or more slices, where each slice consists of an integer number of macroblocks in raster scan order within its slice group. Thus, if a single slice is lost and the remaining slices are successfully reconstructed, several macroblocks that are adjacent to the corrupted macroblocks are available to assist error concealment.

In contrast with previous standards in which the picture type (I, P, or B) determined the macroblock prediction modes available, it is the slice type in H.264/AVC that determines which prediction modes are available for the macroblocks. For example, I slices contain only intra predicted macroblocks, and P slices can contain inter predicted (motion-compensated) macroblocks in addition to the types allowed in I slices. Slices of different types can be mixed within a single picture.

3.2.1.2 Spatial Directional Intra Prediction. The compression performance of a block-based video codec depends fundamentally on the effective use of prediction of sample blocks to minimize the residual prediction error. H.264/AVC provides a very powerful and flexible model for the prediction of each block of samples from previously encoded and reconstructed sample values. This includes spatial directional prediction within the same picture (intra prediction), and flexible multiple reference picture motion compensation from a set of previously reconstructed and stored pictures (inter prediction). As stated above, the prediction modes that are available are dependent upon the type of each slice (intra, predictive, or bipredictive, for I, P, and B slices, respectively).

Intra prediction requires data from only within the current picture. Unlike the previous video standards, in which prediction of intra coded blocks was achieved by predicting only the DC value or a single row or column of transform coefficients from the above or left block, H.264/AVC uses spatial directional prediction, in which individual sample values are predicted based on neighboring sample values that have already been decoded and fully reconstructed. Two different modes are supported: 4×4 and 16×16. In the 4×4 intra coding mode, each 4×4 luma block within a macroblock can use a different prediction mode. There are nine possible modes: DC and eight directional prediction modes. For example, in the horizontal prediction mode, the prediction is formed by copying the samples immediately to the left of the block across the rows of the block. As illustrated in Fig. 14,

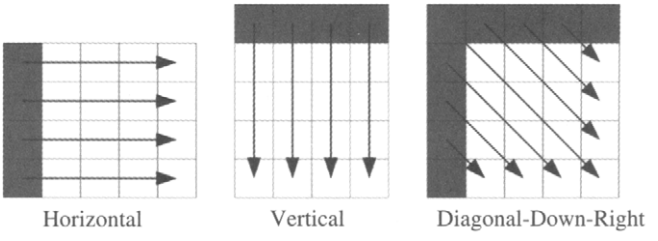


FIGURE 14 Intra prediction in H.264/AVC.

[2]Trick-mode playback includes capabilities such as fast forward and reverse, slow forward and reverse, jumping from section to section, and other forms of unusual navigation through a coded video stream.

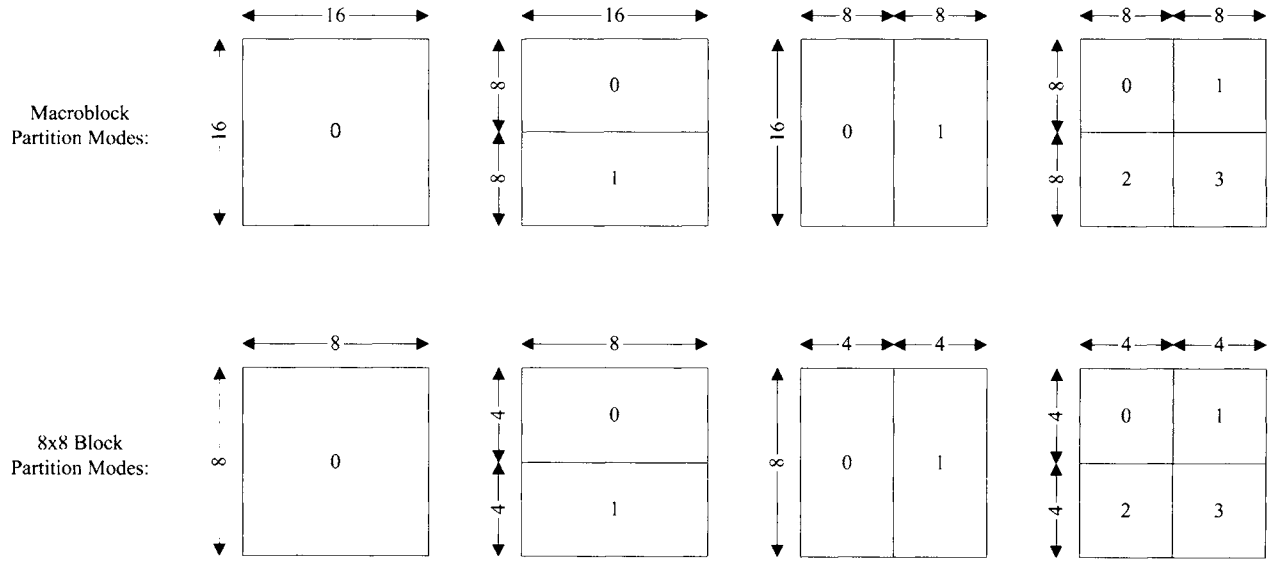


FIGURE 15 Illustration of macroblock partitioning into blocks of different sizes (dimensions shown are in units of luma samples).

diagonal modes operate similarly, with the prediction of each sample based on a weighting of the previously reconstructed samples adjacent to the predicted block.

The 16 × 16 intra prediction mode operates similarly, except that the entire luma macroblock is predicted at once, based on the samples above and to the left of the macroblock. Also, in this mode there are only 4 modes available for prediction: DC, vertical, horizontal and planar. The 16 × 16 intra mode is most useful in relatively smooth picture areas.

3.2.1.3 Enhanced Motion-Compensation Prediction Model. Motion-compensated prediction (inter prediction) in H.264/AVC, similar to that in the previous standards, is primarily based on the translational block-based motion model, in which blocks of samples from previously reconstructed reference pictures are used to predict current blocks through transmission of motion vectors. However, the motion-compensation model defined in H.264/AVC [25, 26] is more powerful and flexible than those defined in earlier standards, and provides a much larger number of options in the search for block matches to minimize the residual error. The H.264/AVC motion model includes seven partition sizes for motion compensation, quarter-sample accurate motion vectors, a generalized multiple reference picture buffer, and weighted prediction. For an encoder to take full advantage of the larger number of prediction options, the prediction selection in H.264/AVC is more computationally intense than in some earlier standards that have simpler models, such as MPEG-2.

Previous standards typically allowed motion compensation block sizes of only 16 × 16 or possibly 8 × 8 luma samples. In H.264/AVC, the partitioning of each macroblock into blocks for motion compensation is much more flexible, allowing seven different block sizes, as illustrated in Fig. 15. Each macroblock can be partitioned in one of the four ways illustrated in the top row of the figure. If the 8 × 8 partitioning is chosen, each 8 × 8 block can be further partitioned in one of the four ways shown in the bottom row of the figure.

All motion vectors in H.264/AVC are transmitted with quarter-luma sample accuracy, providing improved opportunities for prediction compared to most previous standards, which allowed only half-sample accurate motion compensation. For motion vectors at fractional-sample locations, the sample values are interpolated from the reference picture samples at integer-sample locations. Luma component predictions at half-sample locations are interpolated using a 6-tap finite impulse response (FIR) filter. Predictions at quarter-sample luma locations are computed via a bilinear interpolation of the values for two neighboring integer- and half-sample locations. Chroma fractional-sample location values are computed by linear interpolation between integer-location values.

H.264/AVC provides great flexibility in terms of which pictures can be used as references to generate motion-compensated predictions for subsequent coded pictures. This is achieved through a flexible multiple reference picture buffer. In earlier standards, the availability of reference pictures was generally fixed and limited to a single temporally previous picture for predicting P pictures, and two pictures — one temporally previous and one temporally subsequent — for predicting B pictures. However, in H.264/AVC, a multiple-reference picture buffer is available and it may contain up to 16 reference frames or 32 reference fields (depending on the profile, level, and picture resolution specified for the bitstream). The assignment and removal of pictures entering and exiting the buffer can be explicitly controlled by the

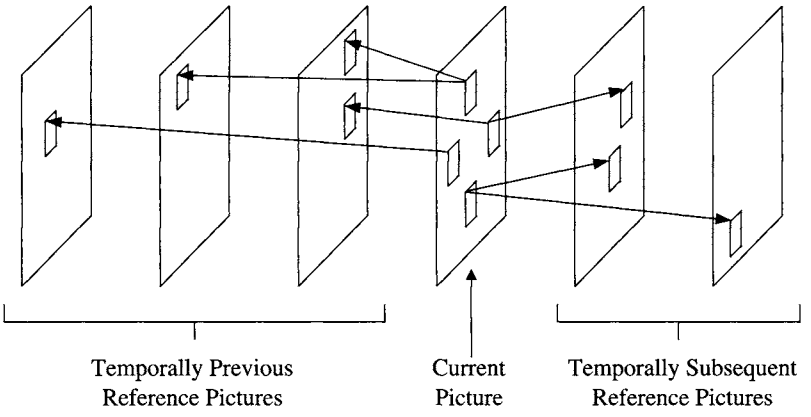


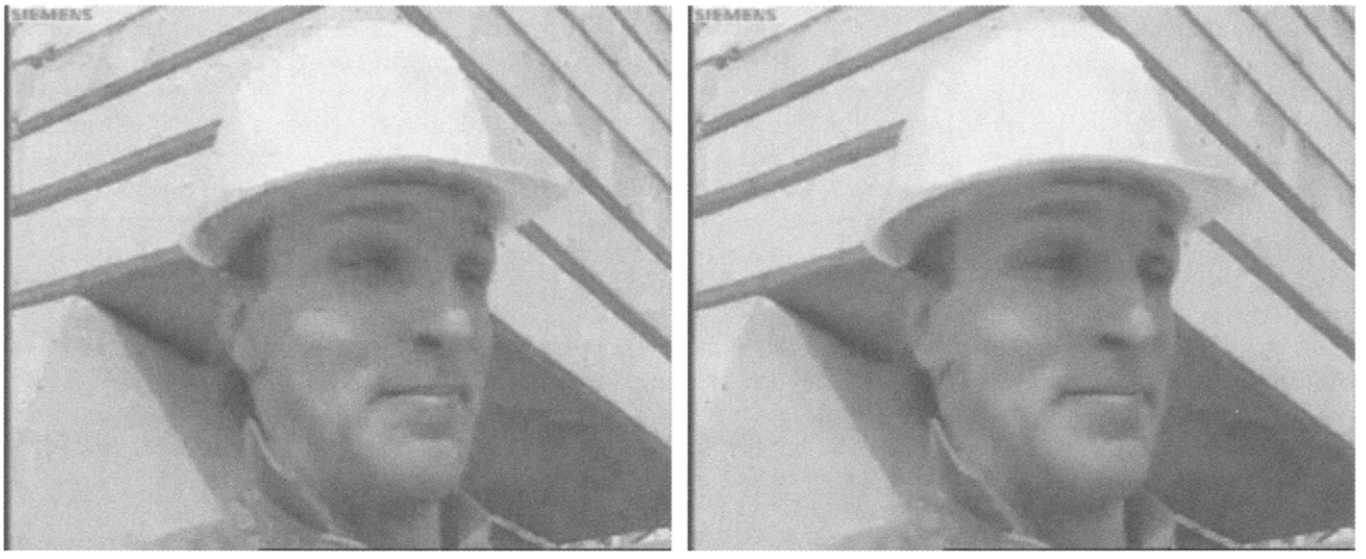
FIGURE 16 Multiple reference picture prediction in H.264/AVC.

encoder by transmitting buffer control commands as side-information in the slice header, or the buffer can operate on a first-in, first-out basis in decoding order. The motion-compensated predictions for each macroblock can be derived from one or more of the reference pictures within the buffer by including reference picture selection syntax elements in conjunction with the motion vectors. With the generalized multiple reference picture buffer, the temporal constraints that are imposed on reference picture usage are greatly relaxed. For example, as illustrated in Fig. 16, it is possible that a reference picture used in a P slice is located temporally subsequent to the current picture, or that both references for predicting a macroblock in a B slice are located in the same temporal direction (either forward or backward in temporal order). To convey the generality, the terms *forward* and *backward* prediction are not used in H.264/AVC. Instead the terms *List 0* prediction and *List 1* prediction are used, reflecting the organization of pictures in the reference buffer into two (possibly overlapping) lists of pictures without regard to temporal order. In addition, with the generalized multiple reference picture buffer, pictures coded using B slices can be referenced for the prediction of other pictures, if desired by the encoder [25].

In P slices, each motion-compensated luma prediction block is derived by quarter-sample interpolation of a prediction from a single block within the set of reference pictures. In B slices, an additional option is provided, allowing each prediction block to be derived from a pair of locations within the set of reference pictures. By default, the final prediction in a bipredictive block is computed using a sample-wise ordinary average of the two interpolated reference blocks. However, H.264/AVC also includes a weighted prediction feature, in which each prediction block can be assigned a relative weighting value rather than using an ordinary average value, and an offset can be added to the prediction. These parameters can either be transmitted explicitly by the encoder, or computed implicitly, based on a time-related relationship between the current picture and the reference picture(s). Weighted prediction can provide substantial coding gains in scenes containing fades or cross fades, which are very difficult to code efficiently with previous video standards that do not include this feature.

3.2.1.4 In-Loop Deblocking Filter. Block-based prediction and transform coding, including quantization of transform coefficients, can lead to visible and subjectively objectionable changes in intensity at coded block boundaries, referred to as blocking artifacts. In previous video codecs, such as MPEG-2, the visibility of these artifacts could optionally be reduced to improve subjective quality by applying a deblocking filter after decoding and prior to display. The goal of such a filter is to reduce the visibility of subjectively annoying artifacts while avoiding excessive smoothing that would result in loss of detail in the image. However, because the process was optional for decoders and was not normatively specified in these standards, the filtering was required to take place outside of the motion-compensation loop, and large blocking artifacts would still exist in reference pictures that would be used for predicting subsequent pictures. This reduced the effectiveness of the motion-compensation process and allowed blocking artifacts to be propagated into the interior of subsequent motion-compensated blocks, causing the subsequent picture predictions to be less effective and making the removal of the artifacts by filtering more challenging.

To improve upon this, H.264/AVC defines a normative deblocking filtering process. This process is performed identically in both the encoder and decoder to maintain an identical set of reference pictures. This filtering leads to both objective and subjective improvements in quality, shown in Fig. 17, due to the improved prediction and the reduction in visible blocking artifacts. Note that the second version of the ITU-T H.263 standard also included an in-loop deblocking filter as an optional feature, but this feature was not supported in the most widely deployed *baseline* profile of that standard, and the design had problems with inverse-transform rounding error effects.



(a) (b)

FIGURE 17 A decoded frame of the sequence *Foreman* (*a*) without the in-loop deblocking filtering applied and (*b*) with the in-loop deblocking filtering (the original sequence *Foreman* is courtesy of Siemens AG) (see color insert).

The deblocking filter defined in H.264/AVC operates on the 4×4 block transform grid. Both luma and chroma samples are filtered. The filter is highly adaptive to remove as many artifacts as possible without excessive smoothing. For the line of samples across each horizontal or vertical block edge (such as that illustrated in Fig. 18), a filtering strength parameter is determined based on the coding parameters on both sides of the edge. When the coding parameters indicate that large artifacts are more likely to be generated (e.g., intra prediction or coding of non-zero transform coefficients), larger strength values are assigned. This results in stronger filtering being applied. Additionally, sample values (i.e., a_0–a_3 and b_0–b_3 in Fig. 18) along each line of samples to be (potentially) filtered are checked against several conditions that are based on the quantization step size used on either side of the edge in order to distinguish between discontinuities that are introduced by quantization, and those that are true edges that should not be filtered to avoid loss of detail [26]. For instance, in the example shown in Fig. 18, a significant lack of smoothness can be detected between the sample values on each side of the block edge. When the quantization step size is large, this lack of smoothness is considered an undesirable artifact and it is filtered. When the quantization step size is small, the lack of smoothness is considered to be the result of actual details in the scene being depicted by the video and it is not altered.

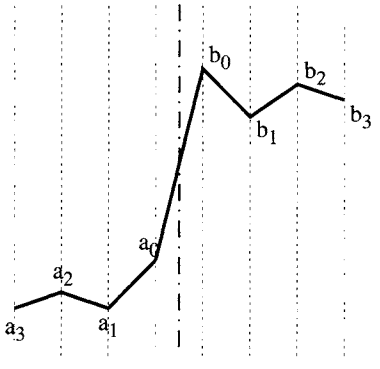


FIGURE 18 Example of an edge profile. Notations a_0, a_1, a_2, a_3, and b_0, b_1, b_2, b_3 stand for sample values on each side of the edge.

3.2.1.5 Transform, Quantization, and Scanning. As in earlier standards, H.264/AVC uses block transform coding to efficiently represent the prediction residual signal. However, unlike previous standards, the H.264/AVC transform is primarily based on blocks of 4×4 samples, instead of 8×8 blocks. Furthermore, instead of performing an approximation of the floating-point DCT (i.e., by specifying that each implementation must choose its own approximation of the ideal IDCT equations), the transform employed has properties similar to that of the 4×4 DCT, but it is completely specified in integer operations. This eliminates the problem of mismatch between the inverse transforms performed in the encoder and decoder, enabling an exact specification of the decoded sample values and significantly reducing the complexity of the IDCT computations. In particular, the IDCT can be computed easily using only 16-bit fixed-point computations (including intermediate values), which would have been very difficult for the IDCT defined in all previous standards. Additionally, the basis functions of the transform are extended by using additional second-stage transforms that are applied to the 2×2 chroma DC coefficients of a macroblock, and the 4×4 luma DC coefficients of a macroblock predicted in the 16×16 intra mode.

Scalar quantization is applied to the transform coefficients [27], but without as wide a dead-zone as used in all other video standards. The 52 quantization parameter values are designed so that the quantization step size increases by approximately 12.2% for each increment of one in the quantization parameter (such that the step size exactly doubles if the quantization parameter is increased by 6). As in earlier standards, each block of quantized coefficients is zig-zag-scanned for ordering the quantized coefficient representations in the entropy coding process. The decoder performs an approximate inversion of the quantization process by multiplying the quantized coefficient values by the step size that was used in their quantization. This inverse quantization process is referred to as scaling in the H.264/AVC standard, since it consists essentially of just a multiplication by a scale factor. In slice types other than SP/SI, the decoder then performs an inverse transform of the scaled coefficients and adds this approximate residual block to the spatial-domain prediction block to form the final picture reconstruction prior to the deblocking filter process.

In SP and SI synchronization slices, the decoder performs an additional forward transform on the prediction block and then quantizes the transform-domain sum of the prediction and the scaled residual coefficients. This additional forward transform and quantization operation can be used by the encoder to discard any differences between the details of the prediction values obtained when using somewhat different reference pictures in the prediction process (for example, differences arising from the reference pictures being from an encoding of the same video content at a different bit rate). The reconstructed result in this case is then formed by an inverse transform of the quantized transform-domain sum, rather than the ordinary case where the reconstructed result is the sum of a spatial-domain prediction and a spatial-domain scaled residual.

3.2.1.6 Entropy Coding. Two methods of entropy coding are specified in H.264/AVC. The simpler variable-length coding method is supported in all profiles. In both cases, many syntax elements except for the quantized transform coefficients are coded using a regular, infinite-extent variable-length codeword set. Here, the same set of Exp-Golomb codewords is used for each syntax element, but the mapping of codewords to decoded values is selected to fit the syntax element.

In the simpler entropy coding method, scans of quantized coefficients are coded using a *context-adaptive variable-length coding* (CAVLC) scheme. In this method, one of a number of variable-length coding tables is selected for each symbol, depending on the contextual information. This includes statistics from previously coded neighboring blocks, as well as statistics from previously coded data for the current block.

For improved coding efficiency, a more complex context-adaptive binary arithmetic coding (CABAC) method can be used [28]. When CABAC is in use, scans of transform coefficients and other syntax elements of the macroblock level and below (such as reference picture indices and motion vector values) are encoded differently. In this method, each symbol is binarized (i.e., converted to a binary code), and then the value of each bin (bit of the binary code) is arithmetically coded. To adapt the coding to non-stationary symbol statistics, context modeling is used to select one of several probability models for each bin, based on the statistics of previously coded symbols. The use of arithmetic coding allows for a non-integer number of bits to be used to code each symbol, which is highly beneficial in the case of very skewed probability distributions. Probability models are updated following the coding of each bit, and this high degree of adaptivity improves the coding efficiency even more than the ability to use a fractional number of bits for each symbol. Due to its bit-serial nature, the CABAC method entails greater computational complexity than the VLC-based method.

3.2.1.7 Frame/Field Adaptive Coding. H.264/AVC includes tools for efficiently handling the special properties of interlaced video, since the two fields that compose an interlaced frame are captured at different instances of time. In areas of high motion, this can lead to less statistical correlation between adjacent rows within the frame, and greater correlation within individual fields, making field coding (where lines from only a single field compose a macroblock) a more efficient option. In addition to regular frame coding in which lines from both fields are included in each macroblock, H.264/AVC provides two options for special handling of interlaced video: field picture coding and macroblock-adaptive field/frame coding (MB-AFF).

Field coding provides the option of coding pictures containing lines from only a single video field (i.e., a picture composed of only top field lines or only bottom field lines). Depending on the characteristics of the frame, an encoder can choose to code each input picture as two field pictures, or as a complete frame picture. In the case of field coding, an alternate zig-zag coefficient scan pattern is used, and individual reference fields are selected for motion compensated prediction. Thus, frame/field adaptivity can occur at the picture level. Field coding is most effective when there is significant motion throughout the video scene for interlaced input video, as in the case of camera panning.

On the other hand, in MB-AFF coding, the adaptivity between frame and field coding occurs at a level known as the *macroblock pair* level. A macroblock pair consists of a region of the frame that has a height of 32 luma samples and width of 16 luma samples and contains two macroblocks. This coding method is most useful when there is significant motion in some parts of an interlaced video frame and little or no

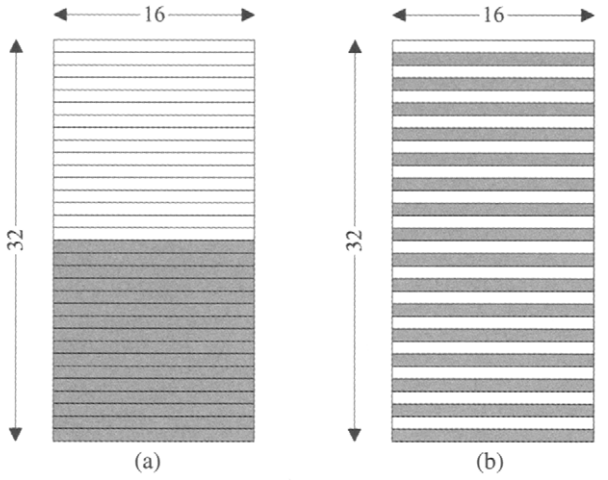


FIGURE 19 Illustration of macroblock pairs.

motion in other parts. For regions with little or no motion, frame coding is typically used for macroblock pairs. In this mode, each macroblock consists of 16 consecutive luma rows from the frame, thus lines from both fields are mixed in the same macroblock. In moving regions, field macroblock pair coding can be used to separate each macroblock pair into two macroblocks that each include rows from only a single field. Frame and field macroblock pairs are illustrated in Fig. 19. The unshaded rows compose the first (or top) macroblock in each pair, and the shaded rows compose the second (or bottom) macroblock. Figure 19(a) shows a frame macroblock pair, in which samples from both fields are combined in each coded macroblock. Figure 19(b) shows a field macroblock pair, in which all of the samples in each of the two macroblocks are derived from a single field. With MB-AFF coding, the internal macroblock coding remains largely the same as in ordinary frame coding or field coding. However, the spatial relationships that are used for motion vector prediction and other context determination become significantly more complicated in order to handle the low-level switching between field and frame based operation.

3.2.2 Profiles

Similar to other standards, H.264/AVC defines a set of profiles that each support only a subset of the entire syntax of the standard and are designed to target specific application areas. The *baseline profile* targets applications not requiring interlace support, and requiring moderate computational complexity, such as videoconferencing. This profile also provides robustness for use on unreliable channels, where some of the video data may be lost or corrupted. The *main profile* targets applications such as broadcast of standard-definition (SD) and high-definition (HD) video on more reliable channels, and storage on optical media. This profile includes more advanced coding tools, such as CABAC entropy coding and B slices, that can improve coding efficiency over that provided by the baseline profile, at the cost of higher complexity. A third profile, called the *extended profile*, is intended for use in streaming and use in error-prone transmission environments, such as wireless networks. The key features that are supported by each of the profiles of H.264/AVC mentioned above are shown[3] in Table 2.

TABLE 2 H.264/AVC profiles

Supported Functionalities	Profile Name		
	Baseline	Main	Extended
I slices	X	X	X
P slices	X	X	X
B slices	—	X	X
SI and SP slices	—	—	X
In-loop deblocking filter	X	X	X
CAVLC entropy decoding	X	X	X
CABAC entropy decoding	—	X	—
Weighted prediction	—	X	X
Field pictures	—	X	X
MB–AFF	—	X	X
Multiple slice groups	X	—	X
Arbitrary slice order	X	—	X
Redundant pictures	X	—	X
Data partitioning	—	—	X

Notation X denotes available functionalities.

3.3 MPEG-4 Compression Performance

In this section, we first present experimental results that compare the coding performance of frame-based coding to object-based coding using an MPEG-4 Part 2 codec on some progressive-scan video content. Next, the coding efficiency of H.264/AVC is compared to that of the prior standards.[4]

3.3.1 MPEG-4 Part 2

While MPEG-4 Part 2 also yields significantly improved coding efficiency over the previous coding standards

[3]Additionally, the JVT has recently finalized work on new profiles of the H.264/AVC standard, which are called the *fidelity range extensions*. These expand the scope of this standard to include enhanced compression capabilities as well as specialized and professional domain applications that require alternative chroma resolution (such as monochrome, 4:2:2, or 4:4:4 chroma sampling) or increased sample accuracy (more than 8 bits per sample). The new profiles are referred to as the high family of profiles — specifically they are the high, high 10, high 4:2:2 and high 4:4:4 profiles.

[4]For space reasons, for each comparison we provide an individual example rather than presenting test results obtained from a large body of experimental data. A proper evaluation of typical or average performance for a given application would require an expanded analysis.

(e.g., 20–30% bit rate savings over MPEG-2 [29]), its main advantage is its object-based representation, which enables many desired functionalities and can yield substantial savings in bit rate savings for some low-complexity video sequences. Here, we present an example that illustrates such a coding efficiency advantage.

The simulations were performed by encoding the color sequence *Bream* at CIF resolution (352 × 288 luma samples/frame) at 10 frames per second (fps) using a constant quantization parameter of 10. The sequence shows a moderate motion scene with a fish swimming and changing directions. We used the MPEG-4 Part 2 reference codec for encoding. The video sequence was coded (a) in frame-based mode (b) in object-based mode at 10 fps. In the frame-based mode, the codec achieved a 56 : 1 compression ratio with relatively high reconstruction quality (34.4 dB). If the quantizer step size were larger, it would be possible to achieve up to a 200 : 1 compression ratio for this sequence, while still keeping the reconstruction quality above 30 dB. In the object-based mode, where the background and foreground (*fish*) objects are encoded separately, a compression ratio of 80 : 1 is obtained. Since the background object did not vary with time, the number of bits spent for its representation was very small. Here, it is also possible to employ sprite coding by encoding the background as a static sprite.

The peak signal-to-noise ratio (PSNR) versus rate performance of the frame-based and object-based coders for the video sequence *Bream* is presented in Fig. 20. As shown in the figure, for this sequence, the PSNR–bit rate tradeoffs of object-based coding are better than those of frame-based coding. This is mainly due to the constant background, which is coded only once in the object-based coding case. However, for scenes with complex and fast varying shapes, since a considerable amount of bits would be spent for shape coding, frame-based coding would achieve better compression levels, but at the cost of a limited object-based access capability.

3.3.2 MPEG-4 Part 10: H.264/AVC

Next, the coding performance of H.264/AVC is compared with that of MPEG-2, H.263 and MPEG-4 Part 2. The results were generated using encoders for each standard that were similarly optimized for rate-distortion performance using Lagrangian coder control. The use of the same efficient rate-distortion optimization method, which is described in [29], allows for a fair comparison of encoders that are compliant with the corresponding standards. Here, we provide one set of results comparing the standards in two key application areas: low-latency video communications and entertainment-quality broadband video.

In the video communications test, we compare the rate-distortion performance of an H.263 *baseline profile* encoder, which is the most widely deployed conformance point for such applications, with an MPEG-4 simple profile encoder and an H.264/AVC baseline profile encoder. The rate-distortion curves generated by encoding the color sequence *Foreman* at CIF resolution (352 × 288 luma samples/frame) at a rate of 15 fps are shown in Fig. 21. As shown in the figure, the H.264/AVC encoder provides significant improvements in coding efficiency. More specifically, H.264/AVC coding yields bit rate savings of approximately 55% over H.263 baseline profile and approximately 35% over MPEG-4 Part 2 simple profile.

A second comparison addresses broadband entertainment-quality applications, where higher resolution content is encoded and larger amounts of latency are tolerated. In this comparison, the H.264/AVC main profile is compared with the widely implemented MPEG-2 main profile. Rate-distortion curves for the interlaced sequence *Entertainment* (720 × 576) are given in Fig. 22. In this plot, we can see that H.264/AVC can yield similar levels of objective quality at approximately half the bit rate for this sequence. In addition to these objective rate-distortion results, extensive subjective testing conducted for the MPEG verification tests of H.264/AVC have

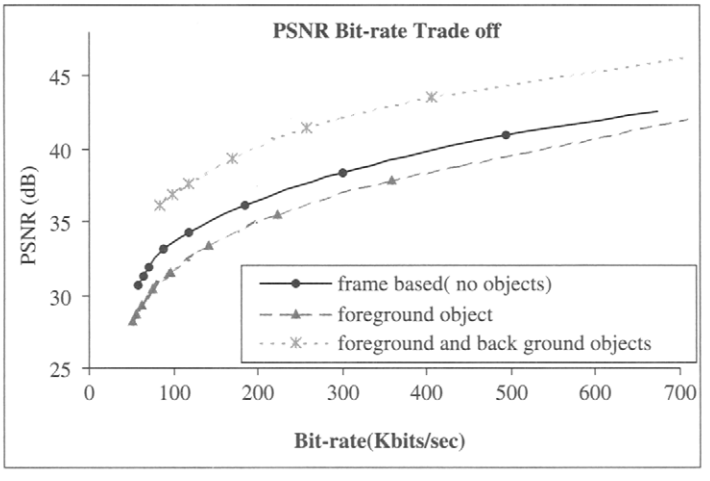


FIGURE 20 Peak signal-to-noise ratio performance for the *Bream* video sequence using different profiles of the MPEG-4 Part 2 video coder.

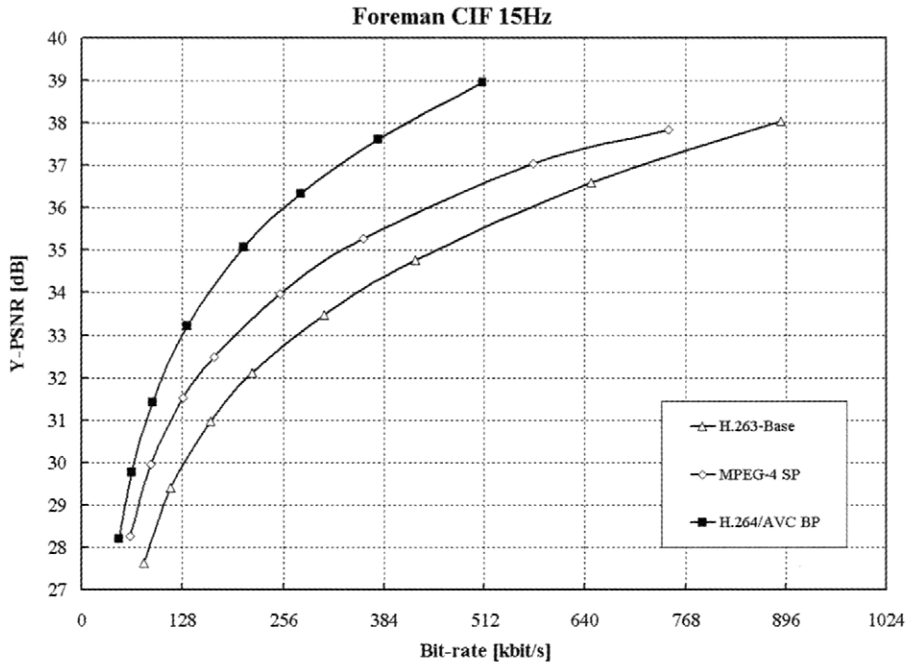


FIGURE 21 Comparison of H.264/AVC baseline profile and the H.263 baseline profile using the sequence *Foreman*.

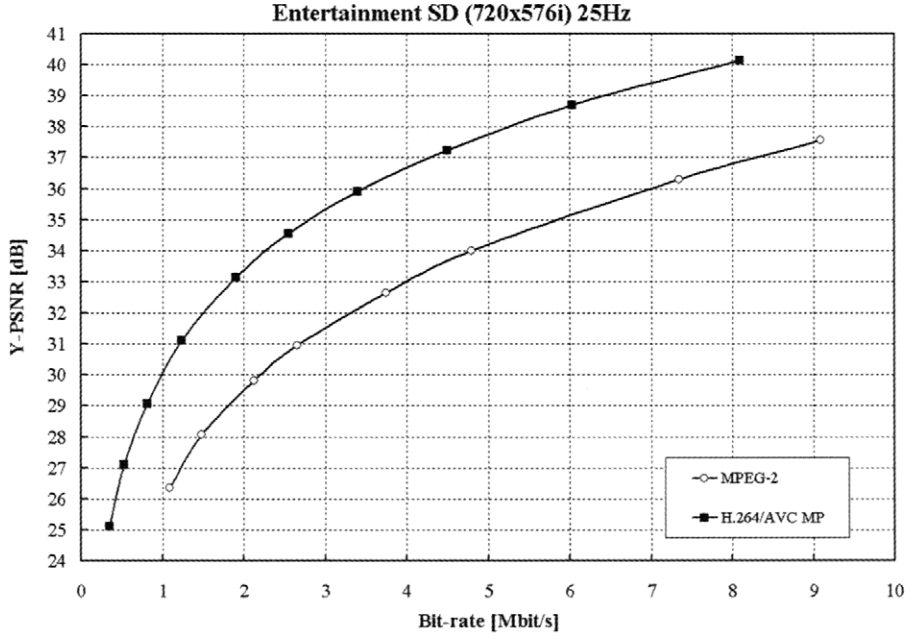


FIGURE 22 Comparison of H.264/AVC main profile and MPEG-2 main profile using the sequence *Entertainment*.

confirmed that similar subjective quality can be achieved with H.264/AVC at approximately half the bit rate of MPEG-2 encoding [30].

3.4 MPEG-4: Video Applications

Although early industry adoption of MPEG-4 Part 2 was significantly slowed down by initial licensing terms that were considered unattractive for some applications, many companies now commercialize a variety of MPEG-4 Part 2 software and hardware products. By far, the most popular implementations of MPEG-4 Part 2 are compliant with just the *simple profile* (SP), and most of these implementations enable video playback in mobile and embedded consumer devices (e.g., cellular phones, personal digital assistants [PDAs], digital cameras, digital camcorders, and personal video recorders).

The MPEG-4 Part 2 *advanced simple profile* (ASP), which permits significantly better video quality than that of SP, has been primarily used in professional video markets, particularly in surveillance cameras and higher quality video coding systems. The availability of two reference software implementations published by the MPEG standards committee and additional sustained efforts to continue providing open source software for MPEG-4 Part 2 may also facilitate the wide adoption of the standard [31].

MPEG-4 Part 10 (H.264/AVC) has rapidly gained support from a wide variety of industry groups worldwide. In the video conferencing industry for example, where video codecs are generally implemented in software, companies such as UB Video have developed H.264/AVC baseline-compliant software codecs that are being already used in many of the popular video conferencing products. Later on, UB Video and other companies such as VideoLocus (now part of LSI Logic) have demonstrated software/hardware products that are compliant with the H.264/AVC main profile. Some of these products are currently being deployed in the broadcast market. Companies such as Apple Computer, Ahead, Polycom, Tandberg, Sony, MainConcept, VideoSoft, KDDI, and others have demonstrated H.264/AVC software products that address a wide market space, from real-time mobile and video conferencing applications to broadcast-quality SD video-coding applications. Also in the broadcast domain, Apple Computer and Modulus Video have demonstrated H.264/AVC real-time software decoding and real-time hardware encoding (respectively) of HD video. Other companies,[5] such as Scientific Atlanta, Tandberg, Envivio, Modulus Video, W&W Communications, Harmonic, DG2L (to mention a few) have demonstrated hardware H.264/AVC encoders and decoders for various broadcast and streaming of sub-SD and SD video. Additionally, the JVT is finalizing a reference software implementation that is publicly available to aid industry deployment and conformance testing efforts.

4 MPEG-7: Technical Description

Digital video compression and coding standards such as MPEG-4 enable the efficient representation of video data, which fuels a rapid increase in the creation and availability of video content. The increased availability of video content creates the need for the efficient search and retrieval of such data. Numerous text-based methods have been applied to the access and manipulation of video content, where keywords are associated with each visual component. Unfortunately, the vocabulary is often restricted, similarity-based retrieval cannot be performed, and human assistance in describing the content and entering the description in the database is required. In order to overcome the limitations of the text-based methods, low level features such as color [32], texture [33], shape [34], and motion [35], combinations of low level features and high level features that capture the structure and semantics of the video data have been employed in the existing content-based access and manipulation (CBAM) systems [36, 37]. As they arise from different applications, these systems make use of various feature representations. For instance, the low level feature *shape* may be represented using Fourier descriptors, region-based representations, etc. Therefore, data accessibility and interoperability between these CBAM systems are quite limited.

MPEG-7, which became an International Standard in 2001, and is officially known as ISO/IEC 15938-3 [38], addresses the above problems by providing a unified framework for multimedia content representation. Unlike its predecessors, which focused mainly on compression, MPEG-7 describes content by standardizing a *multimedia content description interface*. MPEG-7 addresses system issues such as the multiplexing and composition of audiovisual data in the systems part [39], the decoding of the visual data in the visual part [38], and the decoding of the audio data in the audio part [40]. Other MPEG-7 components are described in [41–45]. In this chapter, we will focus on the visual parts of the standard.

MPEG-7 standardizes a set of visual descriptors (Ds), a set of description schemes (DSs) and a description definition language (DDL). In this chapter, we provide an overview of these MPEG-7 video content description tools. MPEG-7 does not standardize the tools that are used to generate the description (e.g., segmentation tools, feature extraction tools) and the tools that use the description (e.g., video content recognition tools). However, the precise definitions of some compliant tools for generating the descriptions are presented in the experimentation model [43, 44].

4.1 Video Content Description

MPEG-7 supports the description of an audiovisual scene such as that illustrated in Fig. 1. The scene description information at the scene and object levels is expressed in terms of color, texture, shape, motion, localization, events and relationships between objects via a set of standardized descriptors, combinations of such descriptors in description schemes and a description definition language. The MPEG-7 descriptions can be represented in text format (XML), binary format (binary format for Metadata, i.e., BiM), or a mixture of the text and binary formats. The transport of the MPEG-7 descriptions, which can be delivered separately from, or jointly with, the audiovisual content that they describe, can take place using various delivery systems, such as MPEG-2 transport streams, Internet protocol (IP), or MPEG-4 streams [56]. The delivery of the MPEG-7 description is outside the scope of the standard.

Audiovisual scenes in an MPEG-7 compliant system are reconstructed and presented by an MPEG-7 terminal at the

[5]Several other companies have announced upcoming H.264/AVC products.

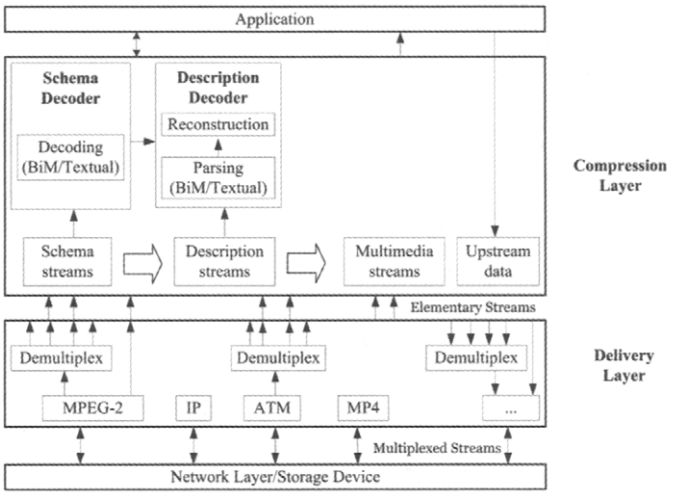


FIGURE 23 Block diagram of an MPEG-7 terminal.

receiver's end. As shown in Fig. 23, an MPEG-7 terminal receives MPEG-7 descriptions from a network or a storage device, demultiplexes the descriptions to retrieve elementary streams, identifies consecutive and individually accessible portions of data known as access units, parses the access units and finally performs the reconstruction of the MPEG-7 description. An MPEG-7 terminal also manages upstream data transfer from the terminal to the transmitter or server. Typically, such data involves queries formulated by the end user or requests for specific information. Once the MPEG-7 descriptions have been reconstructed, they are used by an application separately from, or together with the elementary streams.

4.2 Video Descriptors

For a given video content, a set of features can be extracted. A feature is defined as a distinctive characteristic of the content. In order to compare several features, a meaningful representation of each feature (descriptor) and its instantiation for a given data set (descriptor value) are needed. Fig. 24 illustrates the relationship between data, features and descriptors. Using feature extraction, one projects the input video content space onto the feature space. The result of the projection is a set of features $[f_1, f_2, \ldots, f_i, \ldots, f_N]$ associated with any item of the video content, where N is the total number of features that are extracted. Then, each feature f_i of the feature vector can be represented by several descriptors. For instance, the shape feature may be represented by geometric descriptors or region-based shape descriptors. Those of these descriptors that are standardized in MPEG-7 (i.e., they belong to the standardized descriptor space) are illustrated in Fig. 25.

The projection from the video data space onto the feature space, which is not standardized in MPEG-7, is not unique since different applications may require different features for describing the same video content. The projection from the feature space onto the descriptor space, which is standardized in MPEG-7, is also not unique since several descriptors may be assigned to the same feature. More specifically, several shape descriptors are assigned, for instance, to describe the low-level feature *shape*. This is motivated by various application requirements, that would make use of the entire region shape of the display in our audiovisual scene of Fig. 1, or of the boundary shape of the same display in another application. Because there is no mapping from the descriptor space to the

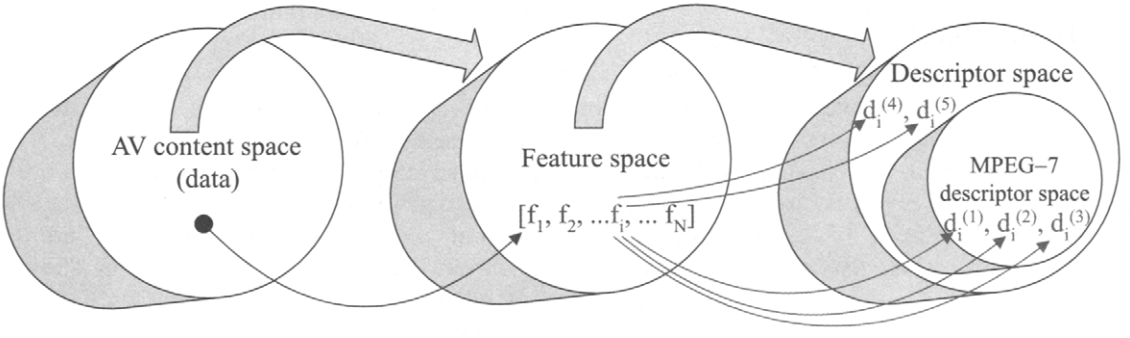


FIGURE 24 The relationship between data, features, and descriptors.

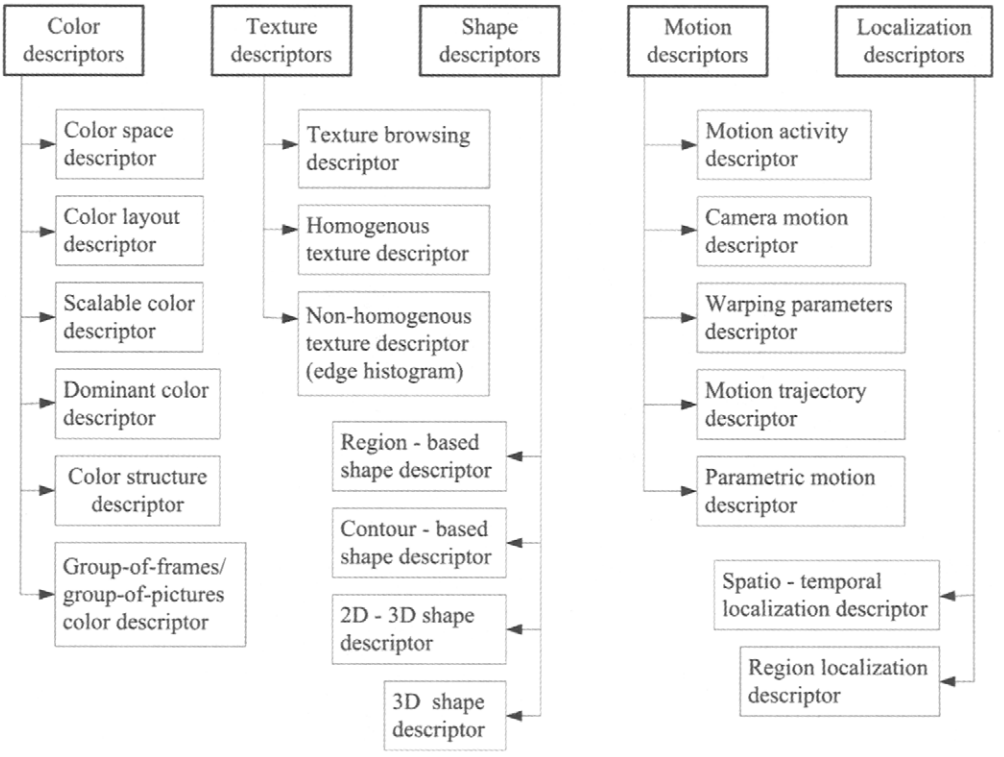


FIGURE 25 Summary of MPEG-7 descriptors.

video content space, the reconstruction of the content using the descriptors is not required. This allows the defining of descriptors that are very compact.

The extraction and usage of MPEG-7 descriptions, which are outside the scope of the standard, are specified in [43, 44]. The usage of MPEG-7 descriptions (i.e., matching), may include feature-to-region, region-to-region, region-to-image, or image-to-image matching. In feature-to-region matching, a texture descriptor instance of a query texture would be compared to a texture descriptor instance associated with the podium in Fig. 1. In region-to-region matching, a texture descriptor instance corresponding to a region in another scene would be compared to a texture descriptor instance of the podium in Fig. 1. To make possible all of these types of comparisons, different types of matching are provided for different descriptors in [43, 44].

4.2.1 Color Descriptors

MPEG-7 standardizes the following descriptors of the low level feature *color*: color space, scalable color, dominant color, color layout, color structure and group-of-frames/group-of-pictures color. Most of the MPEG-7 color descriptors make use of local or global measures of color occurrences that are further assembled in the form of histograms or other structures. Although the color descriptors presented next seem quite similar, they differ in the selection of a color space where each descriptor is computed, in the level of compactness achieved by each descriptor, and in the performance obtained in various MPEG-7 applications. As such, these MPEG-7 color descriptors offer high flexibility for diverse design conditions and types of video content.

Standardized MPEG-7 descriptors for *Color Spaces* include the hue-saturation-value (HSV), red-green-blue (RGB), and YCbCr (luma-chroma) color spaces, and the MPEG-7 defined hue-min-max-dif (HMMD) color space. The latter color space makes use of a hue component (the same as in the HSV color space), a minimum, maximum, and difference components. The minimum component, which is obtained by computing the minimum of the red (R), green (G), and blue (B) components, yields an indication of the amount of *black* in the color. The maximum component, which is obtained by computing the maximum of the R, G, B components, yields an indication of the amount of *white* in the color. The difference component, which is computed by subtracting the minimum component from the maximum component, gives an indication of how much *gray* the color contains.

The *scalable color* descriptor (SCD), which describes the local (region-based) or global (image-based) color distribution in an image, is a color histogram in the HSV color space. The histogram values are non-uniformly quantized coarsely (around 16 bits/histogram) or finely (up to 1,000 bits/histogram) depending on the application. The color histogram is then encoded using a Haar transform [38, 46].

The *dominant color* descriptor (DCD) describes the local or global spatial color distribution feature in any MPEG-7

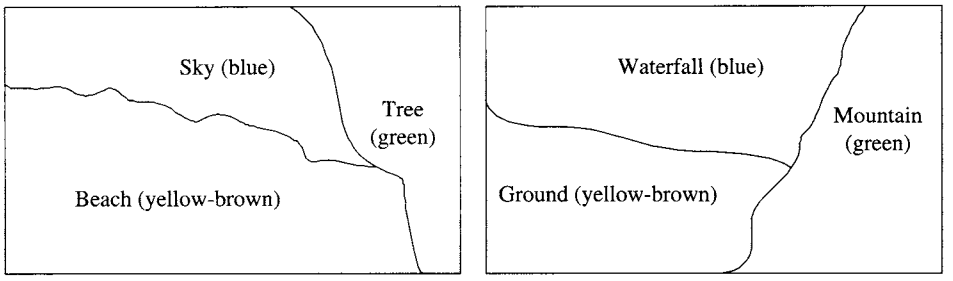


FIGURE 26 Example of color layout similarity.

standardized color space. The descriptor specifies the representative colors, their percentages in a region, the spatial coherency of the color, and the color variance. This descriptor is more compact than the scalable color descriptor and more computationally efficient as it does not require a histogram computation.

The *color layout* descriptor (CLD) describes the spatial distribution of color in an arbitrarily shaped region. This descriptor, which utilizes the YCbCr color space, is computed by dividing an image into 8 × 8 blocks, computing the average of the colors in each block, applying an 8 × 8 discrete cosine transform to each block, and quantizing (for each block) a few low frequency coefficients that are selected by zig-zag scanning. An example of color layout similarity is illustrated Fig. 26. Although the two images are different semantically, their spatial distributions of colors in arbitrarily shaped regions are quite similar.

The *color structure* descriptor (CSD) is obtained by counting the occurrences of a particular color in the image in the hue-min-max-dif color space, and constructing a histogram. The color occurrences are counted within a window sliding over the image. Although the size of the window depends on the image size, the number of samples in the window is maintained constant by subsampling the image and the window simultaneously [38].

The *group-of-frames/group-of-pictures color* descriptor, which describes the color of a collection of images, is an extension of the scalable color descriptor. Consequently, it is computed in the HSV color space. The descriptor consists of the average, median and intersection histograms of groups of images, which are computed using the individual image histograms.

4.2.2 Texture Descriptors

The MPEG-7 texture descriptors aim at capturing the general characteristics of textures in order to describe texture similarity. An example of such texture similarity is illustrated in Fig. 27. The images in Fig. 27(a), which are similar with each other, are irregular textures in the sense that the distribution of texture patterns over the image is not uniform. By contrast, the images in Fig. 27(b), which are also similar with each other, are regular textures. MPEG-7 standardizes three texture descriptors, each of which can describe well the textures in Fig. 27. Because the perception of textures and texture similarity are scale dependent, it is important that the MPEG-7 texture descriptors capture such dependency as well as invariance properties, while maintaining a reasonable representation efficiency.

The *homogeneous texture* descriptor describes the directionality, coarseness, and regularity of patterns in texture images. To obtain this descriptor, the frequency space is first partitioned into 30 channels, which have six divisions that are equal to 30 degrees in the angular direction and five octave divisions in the radial directions. A Radon transform followed by a Fourier transform is applied to the image. Second, to capture the texture scale and orientation, the result of the first step is filtered in the frequency domain using a bank of orientation and scale sensitive filters. Two-dimensional Gabor functions are used to filter the individual feature channels. Third, the energy of every feature channel (i.e., the log-scaled sum of the squared transform coefficients), the energy deviation, the mean and standard deviation of each of the filtered outputs are computed in the frequency domain. The resulting descriptor captures the statistical characteristics of a texture image accurately.

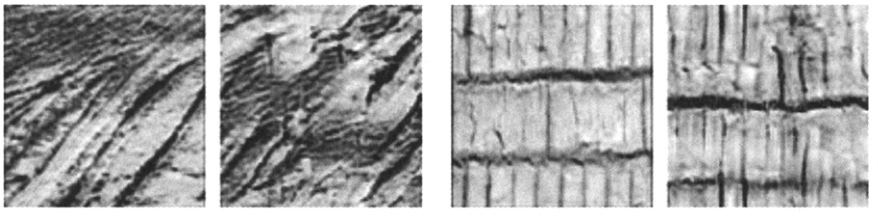


FIGURE 27 Examples of texture similarity.

The *texture browsing* descriptor provides a coarser description of the texture than that obtained using the homogeneous texture descriptor. The computation of the two descriptors is quite similar, however, to achieve a higher compactness that is required for browsing applications, the texture browsing descriptor makes use of only two dominant texture orientations that are extracted from a filtered image. Based on these orientations, six levels of directionality, four levels of coarseness, and four levels of directionality are obtained [38, 46].

The *non-homogeneous texture* descriptor (Edge Histogram descriptor) captures the spatial distribution of edges within an image. This descriptor is computed by dividing the image into 16 non-overlapping blocks of equal size, extracting the edge information and classifying a block as an edge oriented at 0 degrees (horizontal), 90 degrees (vertical), 45 degrees (first diagonal), 135 degrees (second diagonal), or a non-directional edge. Using this classification information, a 5-bin histogram is obtained for each block of the image. The histogram is non-uniformly quantized, yielding a very compact descriptor, which in addition is also scale invariant.

4.2.3 Shape Descriptors

The MPEG-7 shape descriptors allow description of shape objects in terms of region or boundary characteristics. For instance, in our audiovisual scene of Fig. 1, the *desk*, *display*, and *bream* objects would be best described using a region-based shape descriptor, whereas the *funny character* would be best described using a boundary-based shape descriptor. As shown in Fig. 28(a), objects that are similar in terms of the regions covered may be dissimilar in terms of their boundaries. As shown in Fig. 28(b), objects that are similar in terms of their contours may have slightly (or sometimes completely) different covered regions. It is desirable to discriminate between these cases regardless of various transforms (e.g., rotation as shown in Fig. 28B, translation), and shape distortions applied to the subject shapes, segmentation, and feature extraction noise. Even more, in addition to describing region and contour-based shapes in two dimensions, it is desirable to capture the geometric characteristics of 3D objects.

FIGURE 28 Examples of region-based and contour-based similarity.

The *region-based (Angular Radial Transform) Shape* descriptor is obtained using a method that utilizes region-based invariant moments for shape description. For each shape, the method computes the Angular Radial Transform (ART) coefficients on a unit disk in polar coordinates. These coefficients are given by

$$F_{n,m} = \int_0^{2\pi} \int_0^1 V^*_{n,m}(\rho, \theta) f(\rho, \theta)\, \rho\, d\rho\, d\theta \qquad (1)$$

where $V_{n,m}(\rho, \theta)$ is the ART basis function of order n and m, and $f(\rho, \theta)$ is an image intensity function in polar coordinates. The basis functions, which are separable along the radial and angular directions are given by

$$V_{n,m}(\rho, \theta) = \frac{1}{2\pi} \exp(jm\theta)\, R_n(\rho), \qquad (2)$$

where $R_n(\rho) = 1$ for $n = 0$, and $R_n(\rho) = 2cos(\pi n \rho)$ for $n \neq 0$. Twelve angular $(0 \leq m \leq 11)$ and three radial $(0 \leq n \leq 2)$ functions are used, whose real parts are illustrated in Fig. 29 [38]. In this figure, the angular and radial indices m and n are represented on the horizontal and vertical axes (respectively), the origins are in the centers of each image and the bright areas indicate higher values of the functions. The imaginary parts have similar shapes to their corresponding real parts and also have different phases. The coefficients of the ART basis functions are next quantized to ensure the compactness of the descriptor.

The *contour-based shape* descriptor is based on the curvature scale–space (CCS) representation of contours [47]. To obtain a CSS representation, N equidistant points are first

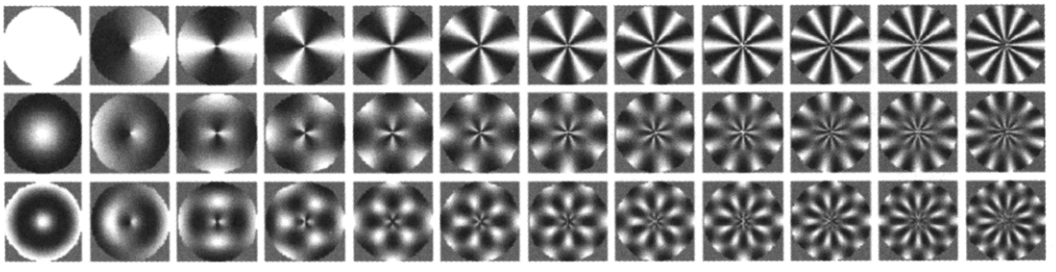


FIGURE 29 The real parts of the angular radial transform basis functions.

selected on the shape boundary starting from an arbitrary point and two series X and Y are formed by grouping the x and y coordinates (respectively). Second, the shape boundary is filtered repeatedly using a lowpass filter and the series of coordinate points X and Y, until it becomes convex. This filtering step requires that the computation of a curvature function of each point of the boundary that was previously selected, computation of the zero crossings of the curvature function, comparison of these zero crossings with those obtained in a previous iteration, and ordering of the zero crossings set, be performed iteratively until there are no more zero crossings. Because the zero crossings of the contour curvature function separate the convex and concave parts of the contour, the absence of zero crossings in the end of the filtering stage indicates that the contour has become convex. Third, by representing graphically the index of each zero crossing during the filtering process and the number of filter passes, a CSS graph is obtained. The peaks are identified in this image, ordered, transformed using a non-linear transform and quantized, yielding a very compact descriptor.

The *2D/3D shape* descriptor describes the shape characteristics of a 3D object in two dimensions. By using a finite number of shapes that are 2D views from different viewing angles of the 3D object, this descriptor makes possible a characterization of the object even when the three dimensional information may not be available. Multiple pairs of such 2D views (one from each object) are needed for similarity evaluation of two 3D objects.

The *3D shape (shape spectrum)* descriptor describes the shape of three dimensional objects using the shape spectrum of a surface. The shape spectrum is defined as the histogram of a shape index, where the shape index is a measure of the local convexity of a surface.

4.2.4 Motion Descriptors

MPEG-7 standardizes four descriptors to represent the motion characteristics in video sequences. Given that the computation of motion fields and optical flow are expensive, a desired property of the motion descriptors is simplicity and efficient computability. Another desired property is compactness, given that motion properties can be described at block, sample and subsample levels.

The *motion activity* descriptor is a global measure of rhythm (also known as *pace*) in a video sequence. Examples of high-activity and low-activity content include sports and video-conferencing sequences, respectively. This descriptor measures the intensity of motion based on standard deviations of motion vector magnitudes. The standard deviations are quantized into five activity values. Optionally, the direction of motion, spatial distribution of motion activity, and the temporal distribution of motion activity can be computed as well and included in the descriptor [38, 46].

The *motion trajectory* descriptor complements the motion activity descriptor in the sense that, the former has a temporal nature, whereas the latter has (most often) a spatial nature. As illustrated in the example of Fig. 30, objects A and B have similar yet not identical trajectories, trajectoryA and trajectoryB, respectively, over the span of N frames of a video sequence. To obtain a good approximation of the trajectory of an object moving spatially and temporally, several keypoints are used, which are computed using first order or second order interpolation models. The interpolation models describe the path of the object between keypoints.

The *camera motion* descriptor describes the global motion in a video sequence. Sometimes, the global motion parameters are provided by the camera. More often than not, such parameters are estimated iteratively using camera motion models. Such models may be generic, from which particular cases of panning, tracking, tilting, booming, zooming, dollying and rolling type of motion are obtained. Using the camera motion descriptor, similar pan sequences can be identified (for instance), in a database of documentary movie sequences. The *warping parameters* descriptor also describes the global motion, however with reference to a mosaic or a sprite such as that illustrated in Fig. 7.

The above motion descriptors characterize motion in terms of generic properties at the video sequence or video segment

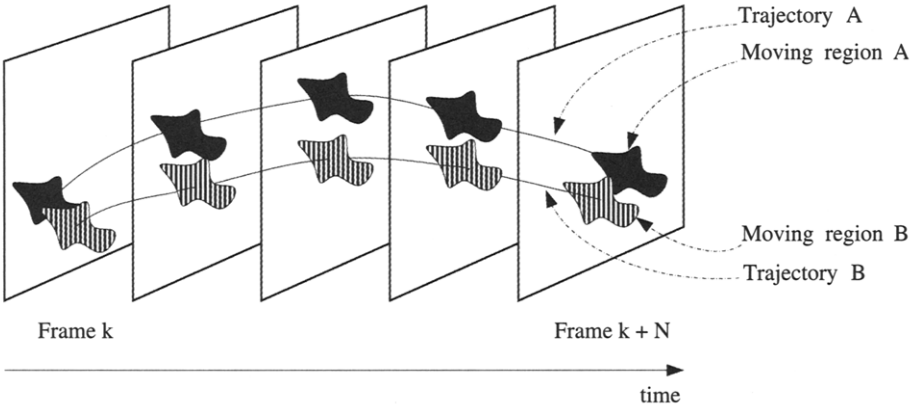


FIGURE 30 Example of motion trajectory similarity.

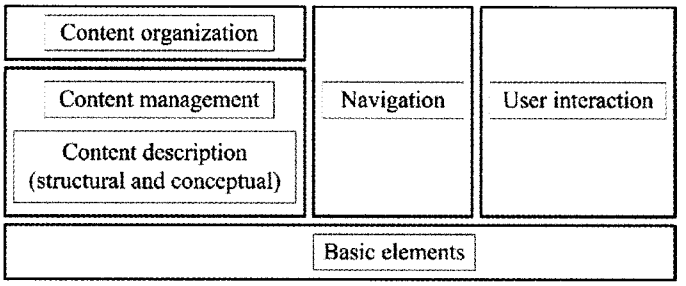


FIGURE 31 Block diagram of a MPEG-7 description scheme.

level. To allow the description of object motion, the *parametric motion* descriptor makes use of a 2D parametric motion (affine, perspective, or quadratic) model [38, 46]. Particular cases of translational, rotational, or other types of motion are obtained using the generic model and make possible the identification of objects that have, for instance, similar rotation motion.

4.2.5 Localization Descriptors

MPEG-7 standardizes two localization descriptors: the *region locator* and the *spatio-temporal locator*. The former descriptor specifies the localization of a region within an image using a representation of a box or a polygon. The latter descriptor specifies the localization of a region in terms of spatial and temporal coordinates.

4.3 Video Description Scheme

An MPEG-7 description scheme (DS), also known as a multimedia description scheme (MDS), is a metadata structure for describing audiovisual content. A description scheme may contain descriptors and other description schemes. As illustrated in Fig. 31, an MPEG-7 description scheme contains components that provide functionalities for content management in terms of creation, production, coding, storage of media, description of structural and semantic aspects of the content, navigation using summaries and hierarchies, organization by classification, and user interaction by user preferences and media usage. These components include basic elements such as data types, constructs for linking media files, localizing content and describing time, places, and organization. In practice, subsets of relevant DSs are selected from the MPEG-7 library of standardized description schemes and combined for specific applications in order to achieve a rich description of the audiovisual content [42, 46]. For instance, in a browsing and navigation application, a hierarchic summary description scheme such as that illustrated in Fig. 32 may be used. The DS contains links to the audiovisual content at the level of segments and frames. The DS can also describe multiple summaries of the same audiovisual content, in order to provide different levels of detail. For example, in a sports video sequence, one HighlightSummary DS may contain the selection of score segments, whereas the other HighlightSummary may contain the arbiter interventions. At the HighlightSegment level, the keyframes corresponding to each of the two classes of highlights would be included.

4.4 Description Definition Language

The description definition language (DDL) is the language used to specify the syntax of the descriptors and description schemes. The DDL also allows the MPEG-7 standardized descriptors and description schemes to be combined, refined

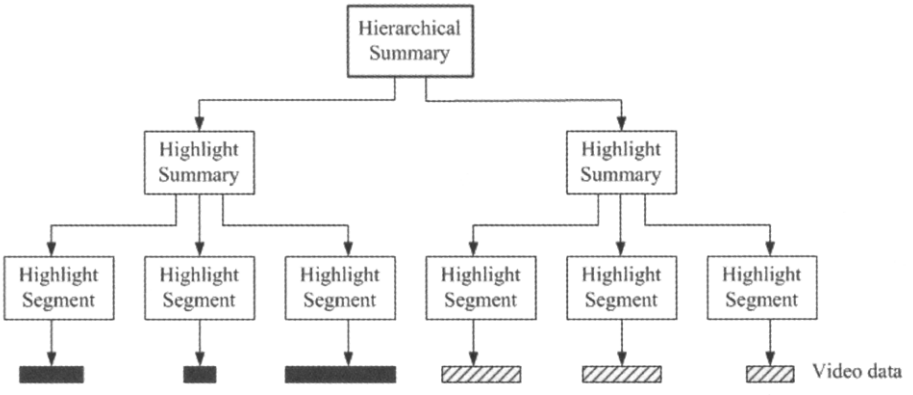


FIGURE 32 Example of a hierarchic summary description scheme.

and extended for specialized applications. For wide interoperability, the DDL is based on the W3C XML Schema Language and it is described in [41] and [45]. Because the XML Schema language has not been designed specifically for audiovisual content, in addition to the XML Schema structural components and data types, the DDL also contains MPEG-7–specific extensions: array and matrix data types, types references, and built-in derived data types [41].

4.5 MPEG-7: Video Applications

Since MPEG-7 became an International Standard, some MPEG-7 conforming video applications have emerged in industry. They include IBM's Annotation Tool, IBM's Automatic Labeling Tool, and Ricoh's MovieTool. The IBM MPEG-7 Annotation Tool (VideoAnnEx) [48] assists in the annotation of video sequences using MPEG-7 metadata. The IBM MPEG-7 Video Labeling tool [49] is an end-to-end MPEG-7 automatic system that performs shot detection, multimodal feature extraction, statistical modeling and semantic concept detection. The Ricoh MovieTool [50] allows the creation of standardized MPEG-7 video content descriptions either automatically during playback or manually by editing the XML file.

5 Conclusions

In this chapter, we have presented a detailed technical description and a brief discussion of some video applications of the visual parts of the two new MPEG standards: MPEG-4 and MPEG-7. MPEG-4, through Part 2 and especially Part 10 (H.264/AVC), enables compliant encoders to achieve high compression levels. MPEG-4 also features object-based, scalability, and error-resilience functionalities that enable both the efficient and robust representation of video data. MPEG-7 does not address the coding efficiency problem, but it also makes use of similar object-based representation and modeling tools to provide complementary functionalities. MPEG-7 can facilitate, and even enable, the object-based access and manipulation of video data by providing a standardized description interface.

Despite some successes, both MPEG-4 Part 2 and MPEG-7 have not yet been as widely adopted in the digital video market as some had hoped. First, MPEG-4 Part 2 has not offered a compelling coding efficiency advantage. Second, the initial MPEG-4 Part 2 licensing terms of MPEG LA, which were considered unattractive for some applications, initially helped dampen enthusiasm towards the wide adoption of MPEG-4 Part 2. It would take several years after the 1999 approval of MPEG-4 Part 2 (in fact nearly up to the time that MPEG-4 Part 10 a.k.a. H.264/AVC was standardized) before MPEG-4 started to become attractive for widespread interoperable use as an alternative to H.263 in the video conferencing and video over IP markets, and to MPEG-2 in the broadcast and streaming (especially HDTV) markets. H.264/AVC has demonstrated very high compression performance, making it the first serious competitor for video codecs that are not compliant with MPEG and ITU-T standards, such as Microsoft's Windows Media 9 and RealNetworks RealVideo. However, H.264/AVC's imminent wide adoption (demonstrated by the fast-growing number of H.264/AVC products) is also the result of licensing terms that have recently been established by the MPEG LA and Via Licensing organizations and that are appropriate for an increasing number of applications. For example, MPEG LA has already negotiated a successful licensing agreement with a group of Japanese broadcasting companies for the adoption of H.264/AVC in their TV applications, and its efforts have also led to the provisional inclusion of H.264/AVC in the emerging HD-DVD standard, which is expected to be widely supported by next-generation HD video playback devices.

Even if H.264/AVC becomes more widely deployed in the digital video market, some other parts of the much larger MPEG-4 standard will remain primarily of academic interest for some time. MPEG-4 Part 2 was developed based on an object-based representation that has not been very practical simply because the segmentation of video content into arbitrarily-shaped, meaningful video objects remains a very difficult, and often practically impossible, task. Fortunately, emerging technologies such as cameras equipped with depth sensors that can capture 3D video and segmented objects [51] may increase the availability of object-based video content, making MPEG-4 Part 2's object-based coding functionality attractive in some applications.

By comparison to MPEG-4, MPEG-7 has had a milder impact on the digital video industry so far. MPEG-7 holds the potential of enabling wide interoperability and efficient access and manipulation of video content. However, the similarly difficult task of extracting low-level and high-level representations from rich video content seems to have stymied the standard's wider applicability to date. For both low-level and high–level representations, automatic extraction is desirable given the large amount of available video content. However, not only automatic spatial and temporal segmentation of natural video content remains a challenge, but even more so automatic or semi-automatic extraction of structure and semantics.

The most experienced image processing researchers agreed at ICIP 1998 (during the 50th anniversary of the IEEE Signal Processing Society) that we, unfortunately, know little about segmentation of, and extraction of structure and semantics from, natural video content. As we learn more about these topics through extensive research work such as [56–58], the adoption of MPEG-4 and MPEG-7 could grow, facilitating and often enabling the efficient representation, access, and manipulation of video data.

References

[1] ISO/IEC 11172-2, "Information Technology—Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbits/s: Video," (1993).

[2] ISO/IEC 13818-2, "Information Technology—Generic coding of moving pictures and associated audio information: Video," (1995).

[3] ITU-T: "Video coding for low bit rate communication," Recommendation H.263 (1996).

[4] ITU-T: "Video coding for low bit rate communication," Recommendation H.263 Version 2 (1998).

[5] Multiple authors, "Special issue on the H.264/AVC video coding standard," *IEEE Trans. Circuits Sys. Video Tech.* 13, (2003).

[6] R. Koenen, "Overview of the MPEG-4 standard," in *ISO/IEC JTC1/SC29/WG11 N4668* (2002).

[7] ISO/IEC 14496-1, "Information technology—Coding of audio-visual objects—Part 1: Systems," (1999).

[8] ISO/IEC 14496-1, "Information technology—Coding of audio-visual objects—Part 2: Visual," (1999).

[9] ISO/IEC 14496-10, "Information technology—Coding of audio-visual objects—Part 10: Advanced video coding," also ITU-T Recommendation H.264: "Advanced video coding for generic audiovisual services," 2003.

[10] ISO/IEC 14496-3, "Information technology—Coding of audio-visual objects—Part 3: Audio," (2001).

[11] MPEG-4 Video Group, "MPEG-4 video verification model version 8.0," ISO/IEC JTC1/SC29/WG11 N3093 (1999).

[12] MPEG-4 SNHC Group, "SNHC verification model 9.0," ISO/IEC JTC1/SC29/WG11 MPEG98/M4116, (1998).

[13] R. M. Haralick and L. H. Shapiro, "Image segmentation techniques," in *Comput. Vis., Graphics Image Process,* 100–132, (1985).

[14] A. K. Katsaggelos, L. P. Kondi, F. W. Meier, J. Ostermann and G. M. Schuster, "MPEG-4 and rate-distortion-based shape-coding techniques," in *Proc. IEEE* 86, 1029–1051 (1998).

[15] M. Lee, W. Chen, C. B. Lin et al. "A layered video object coding system using sprite and affine motion model," in *IEEE Trans. CSVT* 7, 130–146.

[16] M. Tekalp, P. V. Beek, C. Toklu et al. "Two-dimensional mesh-based visual-object representation for interactive synthetic/natural video," in *Proc. IEEE* 86, 1126–1154 (1998).

[17] P. Kalra, A. Mangili, T. N. Magnenat et al. "Simulation of facial muscle actions based on rational free form deformations," in *Proc. IEEE* 86, 1126–1154 (1998).

[18] P. Doenges, T. Capin, F. Lavagetto et al. "MPEG-4: Audio/video and synthetic graphics/audio for real-time, interactive media delivery," in *Image Commun.* 433–463 (1997).

[19] B. G. Haskell, P. G. Howard, Y. A. Lecun et al. "Image and video coding—Emerging standards and beyond," in *Proc. CSVT,* 814–837 (1998).

[20] S. A. Martucci, I. Sodagar, T. Chiang et al. "A zerotree wavelet video coder," in *IEEE Trans. CSVT* 7, 109–118 (1997).

[21] W. Li, "Overview of Fine Granularity Scalability in MPEG-4 Video Standard," in *IEEE Trans. CSVT,* 301–317 (2001).

[22] R. Talluri, "Error Resilient Video Coding in the MPEG-4 Standard," *IEEE Comm. Mag.* 26, 112–119 (1998).

[23] S. Wenger, "H.264/AVC over IP," *IEEE Trans. Circuits Syst. Video Tech.* 13, 645–656 (2003).

[24] T. Stockhammer, M. M. Hannuksela and T. Wiegand, "H.264/AVC in Wireless Environments," *IEEE Trans. Circuits Syst. Video Tech.* 13, 657–673 (2003).

[25] M. Flierl and B. Girod, "Generalized B Pictures and the Draft JVT/H.264 Video Compression Standard," *IEEE Trans. Circuits Syst. Video Tech.* 13, 645–656 (2003).

[26] P. List, A. Joch, J. Lainema et al. "Adaptive Deblocking Filter," *IEEE Trans. Circuits Syst. Video Tech.* 13, 614–619 (2003).

[27] H. S. Malvar, A. Hallapuro and M. Karczewicz, "Low-complexity transform and quantization in H.264/AVC," *IEEE Trans. Circuits Syst. Video Tech.* 13, 598–603 (2003).

[28] D. Marpe, H. Schwarz and T. Wiegand, "Context–adaptive binary arithmetic coding in the H.264/AVC video compression standard," *IEEE Trans. Circuits Syst. Video Tech.* 13, 620–636 (2003).

[29] T. Wiegand, H. Schwarz, A. Joch et al. "Rate-constrained coder control and comparison of video coding standards," *IEEE Trans. Circuits Syst. Video Tech.* 13, 688–703 (2003).

[30] ISO/IEC JTC 1/SC29/WG 11, "Report of the formal verification tests on AVC, document N6231," (2003).

[31] OPENDIVX Open Source MPEG-4 Software," info available at http://www.projectmayo.com.

[32] X. Wan and C. J. Kuo, "A new approach to image retrieval with hierarchical color clustering," *IEEE Trans. Circuits Syst. Video Tech.* 8, 628–643 (1998).

[33] A. K. Jain and K. Karu, "Learning texture discrimination masks," *IEEE Trans. Circuits Syst. Video Tech.* 18, 195–205 (1996).

[34] F. Dell'Acqua and P. Gamba, "Simplified modal analysis and search for reliable shape retrieval," *IEEE Trans. Circuits Syst. Video Tech.* 8, 654–666 (1998).

[35] R. Fablet, P. Bouthemy and P. Perez, "Nonparametric motion characterization using causal probabilistic models for video indexing and retrieval," *IEEE Trans. Circuits Syst. Video Tech.* 11, 393–407 (2002).

[36] IBM, "Query by image content (QBIC) homepage," http://wwwqbic.almaden.ibm.com (1998).

[37] H. R. Naphide and T. S. Huang, "A probabilistic framework for semantic video indexing, filtering and retrieval," *IEEE Trans. Multimedia,* 3, 141–151 (2001).

[38] ISO/IEC 15938-3, "MPEG-7: Visual," (2002).

[39] ISO/IEC 15938-1, "MPEG-7: System," (2002).

[40] ISO/IEC 15938-4, "MPEG-7: Audio," (2002).

[41] ISO/IEC 15938-2, "MPEG-7: Description Definition Language," (2002).

[42] ISO/IEC 15938-5, "MPEG-7: Multimedia Description Schmes," (2002).

[43] ISO/IEC 15938-6, "MPEG-7: Reference Software," (2002).

[44] ISO/IEC 15938-8, "MPEG-7: Extraction and Use of MPEG-7 Description," (2002).

[45] ISO/IEC 15938-10, "MPEG-7: Schema Definition," (2002).

[46] Multiple authors, "Special Issue on MPEG-7," *IEEE Trans. Circuits Syst. Video Tech.* 11, (2001).

[47] F. Mokhtarian and A. K. Mackworth, "A theory of multiscale, curvature-based shape representation for planar curves," *IEEE Trans. Patt. Anal. Mach. Intell.* 14, 789–805 (1992).

[48] IBM, "IBM MPEG-7 Annotation Tool," http://www.alphaworks.ibm.com/tech/videoannex.

[49] C.-Y. Lin, B. L. Tseng, M. Naphade, A. Natsev and J. R. Smith, "VideoAL: A novel end-to-end MPEG-7 video automatic labeling system," in *Proc. Int. Conf. Image Process.* Barcelona, Spain (2003).

[50] Ricoh, "Ricoh Movie Tool," http://www.ricoh.co.jp/src/multimedia/MovieTool/.

[51] 3DV System, "Depth sensor," info available at http://www.3advsystems.com.

[52] S. Santini, A. Gupta and R. Jain, "Emergent semantics through interaction in image databases," *IEEE Trans. Know. Data Eng.* 13, 337–351 (2001).

[53] N. Li and Y. F. Li, "Feature encoding for unsupervised segmentation of color images," *IEEE Trans. Syst. Man Cyber.* 33, 438–447 (2003).

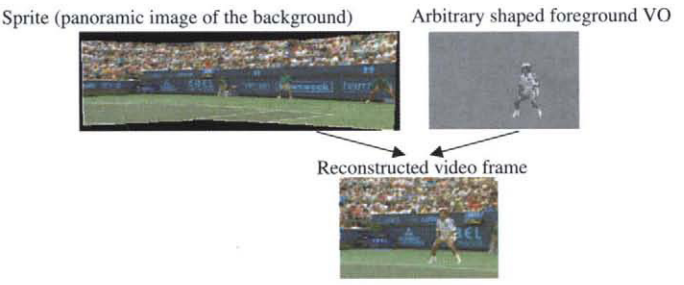


FIGURE 6.5.7 Sprite coding of a video sequence (courtesy of Dr. Thomas Sikora, Technical University of Berlin, [6]). VO, video object.

FIGURE 6.5.17 A decoded frame of the sequence *Foreman* (a) without the in-loop deblocking filtering applied and (b) with the in-loop deblocking filtering (the original sequence *Foreman* is courtesy of Siemens AG).

FIGURE 6.6.1 Embedded video codec aplication in handheld and portable multimedia products.

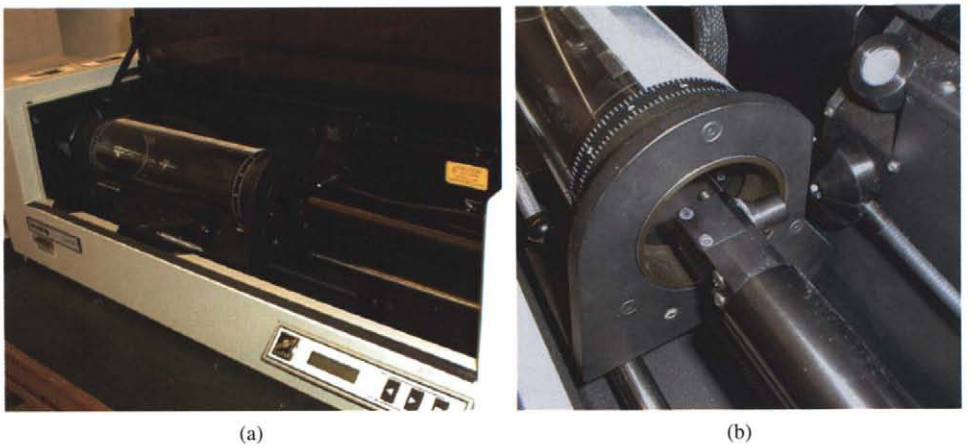


FIGURE 7.1.1 High resolution drum scanner: (a) scanner with cover open, and (b) closeup view showing screw-mounted "C" carriage with light source on inside arm, and detector optics on outside arm.