MPEG-4 Natural Video Coding - An overview

Touradj Ebrahimi* and Caspar Horne**

*Signal Processing Laboratory
Swiss Federal Institute of Technology – EPFL
1015 Lausanne, Switzerland

**Mediamatics Inc.
48430 Lakeview Blvd
Fremont, CA,94538, USA

Abstract

This paper describes the MPEG-4 standard, as defined in ISO/IEC 14496-2. The MPEG-4 visual standard is developed to provide users a new level of interaction with visual contents. It provides technologies to view, access and manipulate objects rather than pixels, with great error robustness at a large range of bit rates. Application areas range from digital television, streaming video, to mobile multimedia and games.
The MPEG-4 natural video standard consists of a collection of tools that support these application areas. The standard provides tools for shape coding, motion estimation and compensation, texture coding, error resilience, sprite coding and scalability.
Conformance points in the form of object types, profiles and levels, provide the basis for interoperability.
Shape coding can be performed in binary mode, where the shape of each object is described by a binary mask, or in gray scale mode, where the shape is described in a form similar to an alpha channel, allowing transparency, and reducing aliasing.
Motion compensation is block based, with appropriate modifications for object boundaries. The block size can be 16x16, or 8x8, with half pixel resolution. MPEG-4 also provides a mode for overlapped motion compensation.
Texture coding is based in 8x8 DCT, with appropriate modifications for object boundary blocks. Coefficient prediction is possible to improve coding efficiency. Static textures can be encoded using a wavelet transform.
Error resilience is provided by resynchronization markers, data partitioning, header extension codes, and reversible variable length codes.
Scalability is provided for both spatial and temporal resolution enhancement. MPEG-4 provides scalability on an object basis, with the restriction that the object shape has to be rectangular.
MPEG-4 conformance points are defined at the Simple Profile, the Core Profile, and the Main Profile. Simple Profile and Core Profiles address typical scene sizes of QCIF and CIF size, with bit rates of 64 kbit/sec, 128 kbit/sec, 384 kbit/sec, and 2Mbit/sec. Main Profile addresses a typical scene sizes of CIF, ITU-R 601, and HD, with bit rates at 2 Mbit/sec, 15 Mbit/sec and 38.4 Mbit/sec.

1 Introduction

Multimedia commands the growing attention of the telecommunications, consumer electronics, and computer industry. In a broad sense, *multimedia* is assumed to be a general framework for interaction with information available from different sources, including video.
A multimedia standard is expected to provide support for a large number of applications. These applications translate into specific sets of requirements which may be very different from each other. One theme common to most applications is the need for supporting interactivity with different kinds of data. Applications related to visual information can be grouped together on the basis of several features:

- type of data (still images, stereo images, video,…);
- type of source (natural images, computer generated images, text/graphics, medical images, …);
- type of communication (ranging from point-to-point to multipoint-to-multipoint);
- type of desired functionalities (object manipulation, on-line editing, progressive transmission, error resilience, …).

Video compression standards, MPEG-1[1] and MPEG-2[2], although perfectly well suited in environments for which they were designed, are not necessarily flexible enough to efficiently address the requirements of multimedia applications. Hence, MPEG (Moving Picture Experts Group) committed itself to the development of the MPEG-4 standard, providing a common platform for a wide range of multimedia applications[3]. MPEG has been working on the development of the MPEG-4 standard since 1993, and finally after about 6 years of efforts, an International Standard covering the first version of MPEG-4 has been adopted recently [4].

This paper provides an overview of the major natural video coding tools and algorithms as defined in the International Standard ISO/IEC 14496-2 (MPEG-4). Additional features and improvements that will be defined in an amendment, due in 2000, are not described here.

2 MPEG-4 visual overview

2.1 Motivation

Digital video is replacing analog video in many existing applications. A prime example is the introduction of digital television that is starting to see wide deployment. Another example is the progressive replacement of analog video cassettes by DVD as the preferred medium to watch movies. MPEG-2 has been one of the key technologies that enabled the acceptance of these new media. In these existing applications, digital video will initially provide similar functionalities as analog video, i.e. the content is represented in digital form instead of analog, with obvious direct benefits such as improved quality and reliability, but the content remains the same to the user. However, once the content is

in the digital domain, new functionalities can easily be added, that will allow the user to view, access, and manipulate the content in completely new ways. The MPEG-4 standard provides key technologies that will enable such functionalities.

2.2 Application areas

2.2.1 Digital TV

With the phenomenal growth of the Internet and the World Wide Web, the interest in advanced interactivity with content provided by digital television is increasing. Increased text, picture, audio, or graphics that can be controlled by the user can add to the entertainment value of certain programs, or provide valuable information unrelated to the current program, but of interest to the viewer. TV station logos, customized advertising, multi-window screen formats allowing display of sports statistics or stock quotes using data-casting are some examples of increased functionalities. Providing the capability to link and to synchronize certain events with video would even improve the experience. Coding and representation of not only frames of video, but also individual objects in the scene (video objects), can open the door for completely new ways of television programming.

2.2.2 Mobile multimedia

The enormous popularity of cellular phones and palm computers indicates the interest in mobile communications and computing. Using multimedia in these areas would enhance the user's experience and improve the usability of these devices. Narrow bandwidth, limited computational capacity, and reliability of the transmission media are limitations that currently hamper wide-spread use of multimedia here. Providing improved error resilience, improved coding efficiency, and flexibility in assigning computational resources would bring mobile multimedia applications closer to reality.

2.2.3 TV production

Content creation is increasingly turning into virtual production techniques as extensions to the well-known chroma keying. The scene and the actors are recorded separately, and can be mixed with additional computer generated special effects. By coding video objects instead of rectangular linear video frames, and allowing access to the video objects, the scene can be rendered with higher quality, and with more flexibility. Television programs consisting of composited video objects, and additional graphics and audio, can then be transmitted directly to the viewer, with the additional advantage of allowing the user to control the programming in a more sophisticated way. In addition, depending on the

targeted viewers, local TV stations could inject regional advertisement video objects, better suited when international programs are broadcast.

2.2.4 Games

The popularity of games on stand-alone game machines, and on PCs clearly indicate the interest in user interaction. Most games are currently using three dimensional graphics, both for the environment, and for the objects that are controlled by the players. The addition of video objects into these games would make the games even more realistic, and using overlay techniques, the objects could be made more life like. Essential is the access to individual video objects, and using standards based technology would make it possible to personalize games by using personal video data bases linked in real-time into the games.

2.2.5 Streaming video

Streaming video over the Internet is becoming very popular, using viewing tools as software plug-ins for a Web browser. News updates and live music shows are just examples of many possible video streaming applications. Here, bandwidth is limited due to the use of modems, and transmission reliability is an issue, as packet loss may occur. Increased error resilience and improved coding efficiency will improve the experience of streaming video. In addition, scalability of the bitstream, in terms of temporal and spatial resolution, but also in terms of video objects, under the control of the viewer, will further enhance the experience, and also the use of streaming video.

2.3 Features and functionalities

The MPEG-4 visual standard consists of a set of tools that enable applications by supporting several classes of functionalities. The most important features covered by MPEG-4 standard can be clustered in three categories (see Fig. 1) and summarized as follows:

1) ***Compression efficiency***: Compression efficiency has been the leading principle for MPEG-1 and MPEG-2, and in itself has enabled applications such as Digital TV and DVD. Improved coding efficiency and coding of multiple concurrent data streams will increase acceptance of applications based on the MPEG-4 standard.

2) ***Content-based interactivity***: Coding and representing video objects rather than video frames enables content-based applications. It is one of the most important novelties offered by MPEG-4. Based on efficient representation of objects, object manipulation, bitstream editing, and object-based scalability allow new levels of content interactivity

3) ***Universal access***: Robustness in error-prone environments allows MPEG-4 encoded content to be accessible over a wide range of media, such as mobile networks as well

as wired connections. In addition, object-based temporal and spatial scalability allow the user to decide where to use sparse resources, which can be the available bandwidth, but also the computing capacity or power consumption.

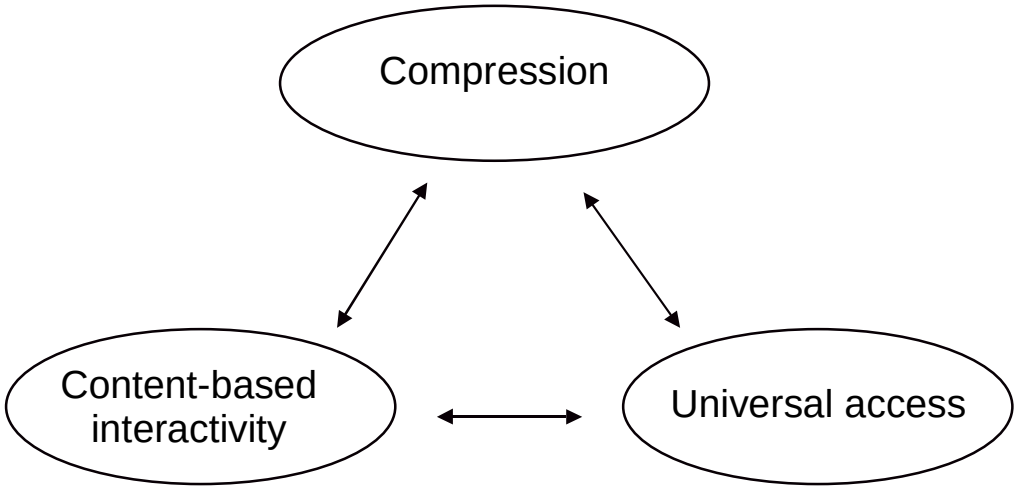


Figure 1: Functionalities offered by the MPEG-4 visual standard

To support some of these functionalities, MPEG-4 should provide the capability to represent arbitrarily shaped video objects. Each object can be encoded with different parameters, and at different qualities. The shape of a video object can be represented in MPEG-4 by a binary or a gray-level (alpha) plane. The texture is coded separately from its shape. For low-bitrate applications, frame based coding of texture can be used, similar to MPEG-1 and MPEG-2. To increase robustness to errors, special provisions are taken into account at the bitstream level to allow fast resynchronization, and efficient error recovery.
The MPEG-4 visual standard has been explicitly optimized for three bitrate ranges:

1) Below 64 kbit/sec
2) 64 - 384 kbit/sec
3) 384- 4 Mbit/sec

For high quality applications, higher bitrates are also supported while using the same set of tools and the same bitstream syntax for those available in the lower bitrates.

MPEG-4 provides support for both interlaced and progressive material.
The chrominance format that is supported is 4:2:0. In this format the number of Cb and Cr samples are half the number of samples of the luminance samples in both horizontal and vertical directions. Each component can be represented by a number of bits ranging from 4 to 12 bits.

2.4 Structure and syntax

The central concept defined by the MPEG-4 standard is the audio-visual object, which forms the foundation of the object-based representation. Such a representation is well suited for interactive applications and gives direct access to the scene contents.
Here we will limit ourselves to mainly natural video objects. However, the discussion remains quite valid for other types of audio-visual objects. A video object may consist of one or more layers to support scalable coding. The scalable syntax allows the reconstruction of video in a layered fashion starting from a standalone base layer, and adding a number of enhancement layers. This allows applications to generate a single MPEG-4 video bitstream for a variety of bandwidth and/or computational complexity requirements. A special case where a high degree of scalability is needed, is when static image data is mapped onto two or three dimensional objects. To address this functionality, MPEG-4 provides a special mode for encoding static textures using a wavelet transform.

An MPEG-4 visual scene may consist of one or more video objects. Each video object is characterized by temporal and spatial information in the form of shape, motion, and texture. For certain applications video objects may not be desirable, because of either the associated overhead or the difficulty of generating video objects. For those applications, MPEG-4 video allows coding of rectangular frames which represent a degenerate case of an arbitrarily shaped object.

An MPEG-4 visual bitstream provides a hierarchical description of a visual scene as shown in Fig. 2. Each level of the hierarchy can be accessed in the bitstream by special code values called start codes. The hierarchical levels that describe the scene most directly are:

- ***Visual Object Sequence (VS):*** The complete MPEG-4 scene which may contain any 2-D or 3-D natural or synthetic objects and their enhancement layers.
- ***Video Object (VO):*** A video object corresponds to a particular (2-D) object in the scene. In the most simple case this can be a rectangular frame, or it can be an arbitrarily shaped object corresponding to an object or background of the scene.

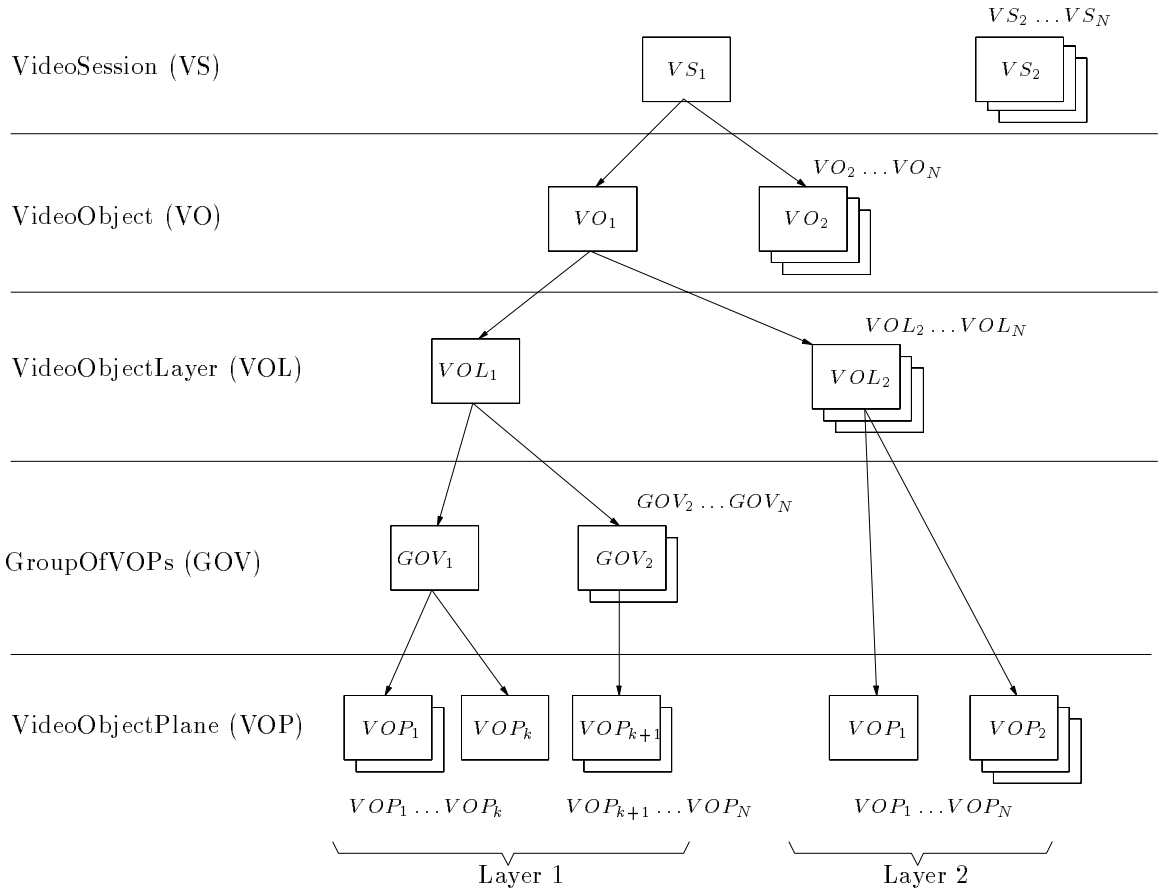


Figure 2: Example of an MPEG-4 video bitstream logical structure

- ***Video Object Layer (VOL)***: Each video object can be encoded in scalable (multi-layer) or non-scalable form (single layer), depending on the application, represented by the video object layer (VOL). The VOL provides support for scalable coding. A video object can be encoded using spatial or temporal scalability, going from coarse to fine resolution. Depending on parameters such as available bandwidth, computational power, and user preferences, the desired resolution can be made available to the decoder.

There are two types of video object layers, the video object layer that provides full MPEG-4 functionality, and a reduced functionality video object layer, the video object layer with short headers. The latter provides bitstream compatibility with base-line H.263 [5].

Each video object is sampled in time, each time sample of a video object is a video object plane. Video object planes can be grouped together to form a group of video object planes:

- ***Group of Video Object Planes (GOV):*** The GOV groups together video object planes. GOVs can provide points in the bitstream where video object planes are

encoded independently from each other, and can thus provide random access points into the bitstream. GOVs are optional.

- ***Video Object Plane (VOP):*** A VOP is a time sample of a video object. VOPs can be encoded independently of each other, or dependent on each other by using motion compensation. A conventional video frame can be represented by a VOP with rectangular shape.

A video object plane can be used in several different ways. In the most common way the VOP contains the encoded video data of a time sample of a video object. In that case it contains motion parameters, shape information and texture data. These are encoded using macroblocks. It can also be used to code a sprite. A sprite is a video object that is usually larger than the displayed video, and is persistent over time. There are ways to slightly modify a sprite, by changing its brightness, or by warping it to take into account spatial deformation. It is used to represent large, more or less static areas, such as backgrounds. Sprites are encoded using macroblocks.

A macroblock contains a section of the luminance component and the spatially subsampled chrominance components. In the MPEG-4 visual standard there is support for only one chrominance format for a macroblock, the 4:2:0 format. In this format, each macroblock contains 4 luminance blocks, and 2 chrominance blocks. Each block contains 8x8 pixels, and is encoded using the DCT transform. A macroblock carries the shape information, motion information, and texture information.

Figure 3 illustrates the general block diagram of MPEG-4 encoding and decoding based on the notion of video objects. Each video object is coded separately. For reasons of efficiency and backward compatibility, video objects are coded via their corresponding video object planes in a hybrid coding scheme somewhat similar to previous MPEG standards. Figure 4 shows an example of decoding of a VOP.

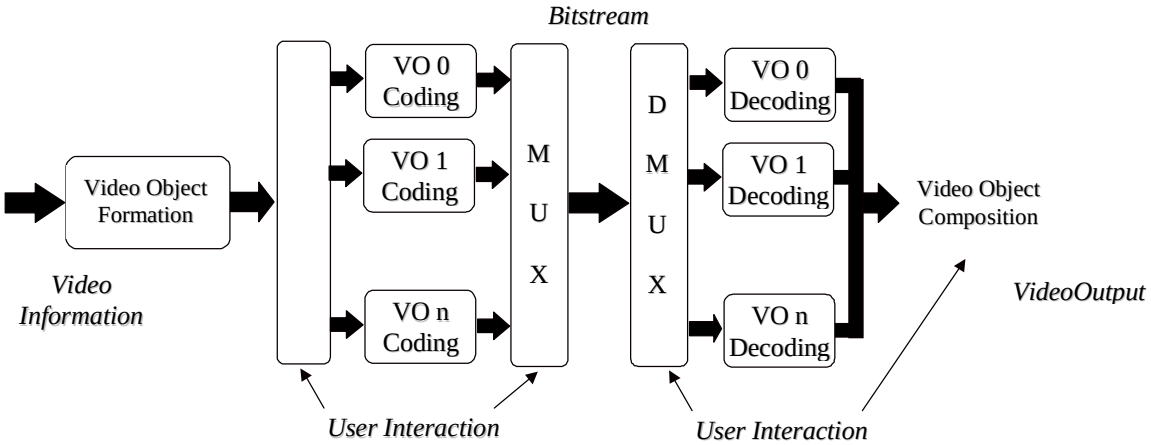


Figure 3: General block diagram of MPEG-4 video

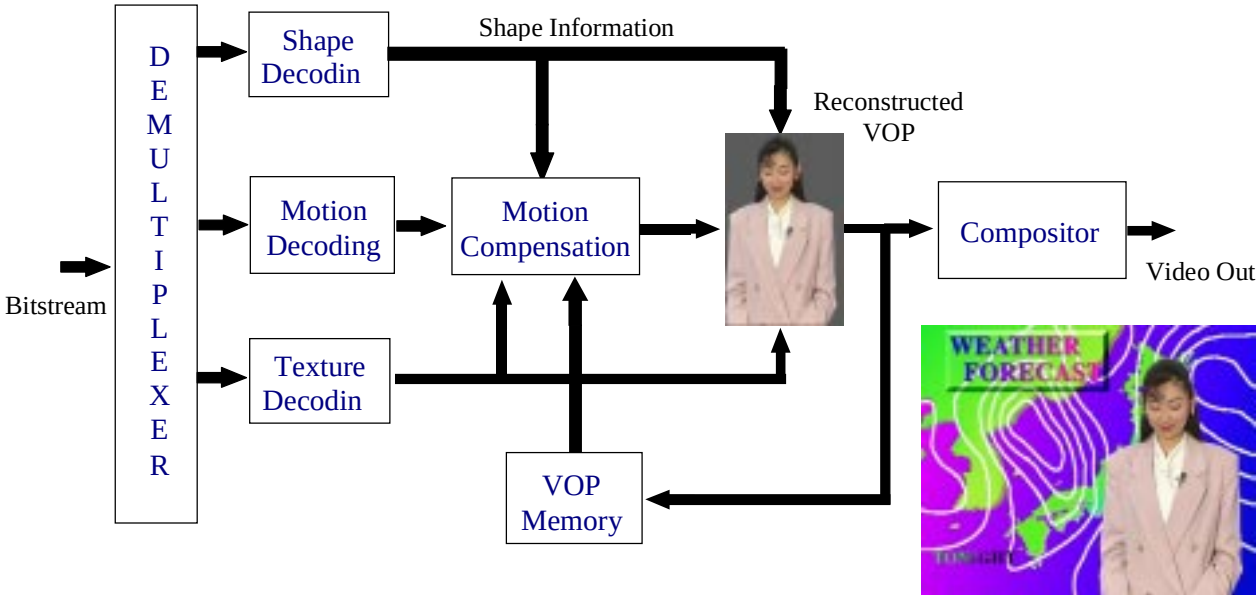


Figure 4: Example of VOP based decoding in MPEG-4

3 Shape coding tools

In this section, we discuss the tools offered by the MPEG-4 standard for explicit coding of shape information for arbitrarily shaped VOs. Besides the shape information available for the VOP in question, the shape coding scheme also relies on motion estimation to compress the shape information even further. A general description of shape coding techniques would be out of the scope of this paper. Therefore, we will only describe the scheme adopted by MPEG-4 natural video standard for shape coding. Interested readers are referred to [6] for information on other shape coding techniques.

In MPEG-4 visual standard, two kinds of shape information are considered as inherent characteristics of a video object. These are referred to as binary and gray scale shape information. By binary shape information, one means a label information that defines which portions (pixels) of the support of the object belong to the video object at a given time. The binary shape information is most commonly represented as a matrix with the same size as that of the bounding box of a VOP. Every element of the matrix can take one of the two possible values depending on whether the pixel is inside or outside the

video object. Gray scale shape is a generalization of the concept of binary shape providing a possibility to represent transparent objects, and reduce aliasing effects. Here the shape information is represented by 8 bits, instead of a binary value.

3.1 Binary shape coding

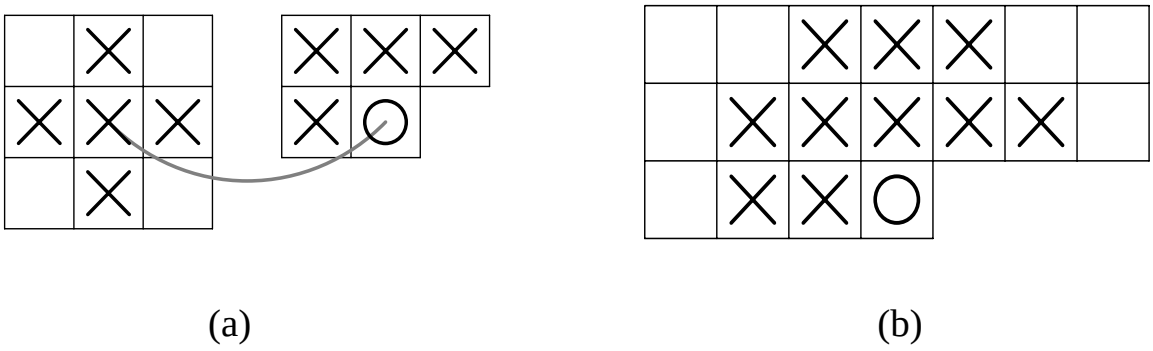
In the past, the problem of shape representation and coding has been thoroughly investigated in the fields of computer vision, image understanding, image compression and computer graphics. However, this is the first time that a video standardization effort has adopted a shape representation and coding technique within its scope.
In its canonical form, a binary shape is represented as a matrix of binary values called a bitmap. However, for the purpose of compression, manipulation, or a more semantic description, one may choose to represent the shape in other forms such as using geometric representations or by means of its contour.
Since its beginning, MPEG adopted a bitmap based compression technique for the shape information. This is mainly due to the relative simplicity and higher maturity of such techniques. Experiments have shown that bitmap based techniques offer good compression efficiency with relatively low computational complexity.
This section describes the coding methods for binary shape information. Binary shape information is encoded by a motion compensated block based technique allowing both lossless and lossy coding of such data. In MPEG-4 video compression algorithm, the shape of every VOP is coded along with its other properties (texture and motion). To this end, the shape of a VOP is bounded by a rectangular window with a size of multiples of 16 pixels in horizontal and vertical directions. The position of the bounding rectangle could be chosen such that it contains the minimum number of blocks of size 16x16 with non transparent pixels. The samples in the bounding box and outside of the VOP are set to 0 (transparent). The rectangular bounding box is then partitioned into blocks of 16x16 samples and the encoding/decoding process is performed block by block.

The binary matrix representing the shape of a VOP is referred to as *binary mask*. In this mask every pixel belonging to the VOP is set to 255, and all other pixels are set to 0. It is then partitioned into *binary alpha blocks* (BAB) of size 16x16. Each BAB is encoded separately. Starting from rectangular frames, it is common to have BABs which have all pixels of the same value, either 0 (in which case the BAB is called a *transparent* block) or 255 (in which case the block is said to be an *opaque* block). The shape compression algorithm provides several modes for coding a BAB. The basic tools for encoding BABs are the Context based Arithmetic Encoding (CAE) algorithm[7] and motion compensation. InterCAE and IntraCAE are the variants of the CAE algorithm used with and without motion compensation, respectively. Each shape coding mode supported by the standard is a combination of these basic tools. Motion vectors can be computed by searching for a best match position (given by the minimum sum of absolute difference). The motion vectors themselves are differentially coded. Every BAB can be coded in one of the following modes:

1. The block is flagged transparent. In this case no coding is necessary. Texture information is not coded for such blocks either.
2. The block is flagged opaque. Again, shape coding is not necessary for such blocks, but texture information needs to be coded (since they belong to the VOP).
3. The block is coded using IntraCAE without use of past information.
4. Motion vector difference (MVD) is zero but the block is not updated.
5. MVD is zero and the block is updated. InerCAE is used for coding the block update.
6. MVD is non-zero, but the block is not coded.
7. MVD is non-zero, and the block is coded.

The CAE algorithm is used to code pixels in BABs. The arithmetic encoder is initialized at the beginning of the process. Each pixel is encoded as follows:

1. Compute a context number according to the definition of Fig. 5;
2. Index a probability table using this context number
3. Use the retrieved probability to drive the arithmetic encoder for codeword assignment.



(a) (b)

Figure 5: Context number selected for InterCAE (a) and IntraCAE (b) shape coding. In each case, the pixel to be encoded is marked by a circle, and the context pixels are marked with crosses. In the InterCAE, part of the context pixels are taken from the co-located block in the previous frame.

3.2 Gray scale shape coding

The gray scale shape information has a similar corresponding structure to that of binary shape with the difference that every pixel (element of the matrix) can take on a range of values (usually 0 to 255) representing the degree of the transparency of that pixel. The gray scale shape corresponds to the notion of alpha plane used in computer graphics, in which 0 corresponds to a completely transparent pixel and 255 to a completely opaque pixel. Intermediate values of the pixel correspond to intermediate degrees of transparencies of that pixel. By convention, a binary shape information corresponds to a gray scale shape information with values of 0 and 255.

Gray scale shape information is encoded using a block based motion compensated DCT similar to that of texture coding, allowing lossy coding only. The gray scale shape coding also makes use of binary shape coding for coding of its support.

4 Motion estimation and compensation tools

Motion estimation and compensation are commonly used to compress video sequences by exploiting temporal redundancies between frames. The approaches for motion compensation in the MPEG-4 standard are similar to those used in other video coding standards. The main difference is that the block-based techniques used in the other standards have been adapted to the VOP structure used in MPEG-4. MPEG-4 provides three modes for encoding an input VOP, as shown in Fig. 6, namely:

1. A VOP may be encoded independently of any other VOP. In this case the encoded VOP is called an *Intra VOP* (I-VOP).
2. A VOP may be predicted (using motion compensation) based on another previously decoded VOP. Such VOPs are called *Predicted VOPs* (P-VOP).
3. A VOP may be predicted based on past as well as future VOPs. Such VOPs are called *Bidirectional Interpolated VOPs* (B-VOP). B-VOPs may only be interpolated based on I-VOPs or P-VOPs.

Obviously, motion estimation is necessary only for coding P-VOPs and B-VOPs. Motion estimation (ME) is performed only for macroblocks in the bounding box of the VOP in question. If a macroblock lies entirely within a VOP, motion estimation is performed in the usual way, based on block matching of 16 x 16 macroblocks as well as 8 x 8 blocks (in advanced prediction mode). This results in one motion vector for the entire macroblock, and one for each of its blocks. Motion vectors are computed to half-sample precision.

For macroblocks that only partially belong to the VOP, motion vectors are estimated using the *modified block (polygon) matching* technique. Here, the discrepancy of matching is given by the sum of absolute difference (SAD) computed for only those pixels in the macroblock that belong to the VOP. In case the reference block lies on the VOP boundary, a repetitive padding technique assigns values to pixels outside the VOP. The SAD is then computed using these padded pixels as well. This improves efficiency, by allowing more possibilities when searching for candidate pixels for prediction at the boundary of the reference VOP.

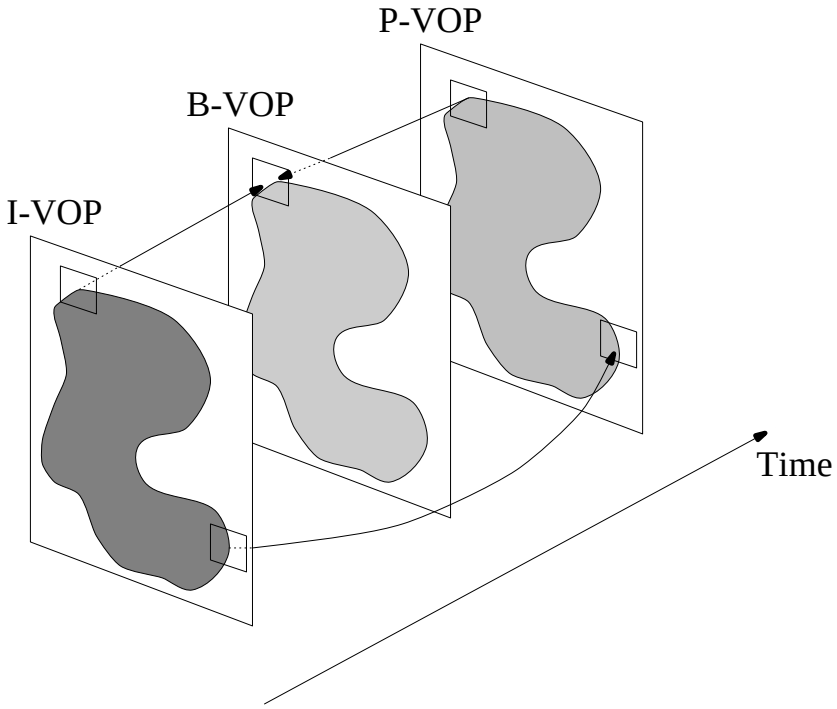


Figure 6: The three modes of VOP coding. I-VOPs are coded without any information from other VOPs. P- and B-VOPs are predicted based on I- or other P-VOPs.

For P- and B-VOPs, motion vectors are encoded as follows. The motion vectors are first differentially coded, based on up to three vectors of previously transmitted blocks. The exact number depends on the allowed range of the vectors. The maximum range is selected by the encoder and transmitted to the decoder, in a fashion similar to MPEG-2. Variable length coding is then used to encode the motion vectors.

MPEG-4 also supports overlapped motion compensation, similar to the one used in H.263 [5]. This usually results in better prediction quality at lower bitrates.
Here, for each block of the macroblock, the motion vectors of neighboring blocks are considered. This includes the motion vector of the current block, and its four neighbors. Each vector provides an estimate of the pixel value. The actual predicted value is then a weighted average of all these estimates.

5 Texture coding tools

The texture information of a video object plane is present in the luminance, Y, and two chrominance components, Cb, Cr, of the video signal. In the case of an I-VOP, the texture information resides directly in the luminance and chrominance components. In the case of motion compensated VOPs the texture information represents the residual error

remaining after motion compensation. For encoding the texture information, the standard 8x8 block-based DCT is used. To encode an arbitrarily shaped VOP, an 8x8 grid is super-imposed on the VOP. Using this grid, 8x8 blocks that are internal to VOP are encoded without modifications. Blocks that straddle the VOP are called boundary blocks, and are treated differently from internal blocks. The transformed blocks are quantized, and individual coefficient prediction can be used from neighboring blocks to further reduce the entropy value of the coefficients. This is followed by a scanning of the coefficients, to reduce to average run length between to coded coefficients. Then, the coefficients are encoded by variable length encoding. This process is illustrated in the block diagram of Fig. 7.

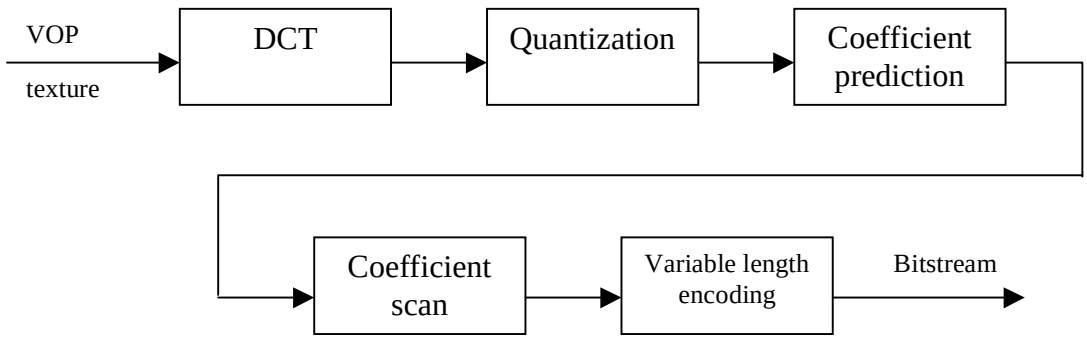


Figure 7: VOP texture coding process

5.1 Boundary macroblocks

Macroblocks that straddle VOP boundaries, boundary macroblocks, contain arbitrarily shaped texture data. A padding process is used to extend these shapes into rectangular macroblocks. The luminance component is padded on 16x16 basis, while the chrominance blocks are padded on 8x8 basis. Repetitive padding consists in assigning a value to the pixels of the macroblock that lie outside of the VOP. When the texture data is the residual error after motion compensation, the blocks are padded with zero-values. Padded macroblocks are then coded using the technique described above.

For intra coded blocks, the padding is performed in a two-step procedure called *Low Pass Extrapolation (LPE)*. This procedure is as follows:

1. Compute the mean of the pixels in the blocks that belong to the VOP. Use this mean value as the padding value, that is,

$$f_{r,c}\Big|_{(r,c)\notin VOP} = \frac{1}{N}\sum_{(x,y)\in VOP} f_{x,y} \qquad (1)$$

where N is the number of pixels of the macroblock in the VOP. This is also known as *mean-repetition* DCT.

2. Use the average operation given in Equ. 2 for each pixel $f_{r,c}$, where r and c are the row and column position of each pixel in the macroblock outside the VOP boundary. Start from the top left corner $f_{0,0}$ of the macroblock and proceed row by row to the bottom right pixel.

$$f_{r,c}\Big|_{(r,c)\notin VOP} = \frac{f_{r,c-1} + f_{r-1,c} + f_{r,c+1} + f_{r+1,c}}{4} \qquad (2)$$

The pixels considered in the right-hand side of Equ. 2 should lie within the VOP, otherwise they are not considered and the denominator is adjusted accordingly.

Once the block has been padded, it is coded in a similar fashion to an internal block.

5.2 DCT

Internal video texture blocks and padded boundary blocks are encoded using a 2-D 8x8 block-based DCT. The accuracy of the implementation of 8x8 inverse transform is specified by the IEEE 1180 standard to minimize accumulation of mismatch errors. The DCT transform is followed by a quantization process.

5.3 Quantization

The DCT coefficients are quantized as a lossy compression step. There are two types of quantizations available. Both are essentially a division of the coefficient by a quantization step size. The first method uses one of two available quantization matrices to modify the quantization step size depending on the spatial frequency of the coefficient. The second method uses the same quantization step size for all coefficients. MPEG-4 also allows for a non-linear quantization of DC values.

5.4 Coefficient prediction

The average energy of the quantized coefficients can be further reduced by using prediction from neighboring blocks. The prediction can be performed from either the block above, the block to the left, or the block above left, as illustrated in Fig. 8. The direction of the prediction is adaptive and is selected based on comparison of horizontal and vertical DC gradients (increase or reduction in its value) of surrounding blocks A, B, and C.

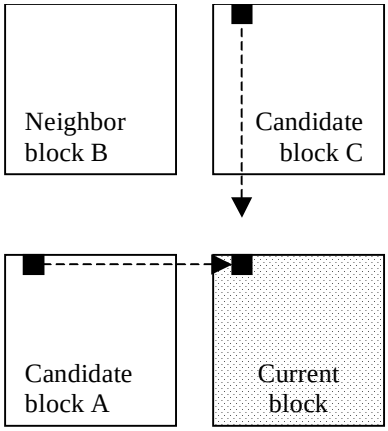


Figure 8: Candidate blocks for coefficient prediction

There are two types of prediction possible, DC prediction and AC prediction:

- ***DC prediction:*** The prediction is performed for the DC coefficient only, and is either from the DC coefficient of block A, or from the DC coefficient of block C.
- ***AC prediction:*** Either the coefficients from the first row, or the coefficients from the first column of the current block are predicted from the co-sited coefficients of the selected candidate block. Differences in the quantization of the selected candidate block are accounted for by appropriate scaling by the ratio of quantization step sizes.

5.5 Scanning and run length coding

Before the coefficients are run length encoded, a scanning process is applied to transform the two dimensional data in a one dimensional string. Three different scanning methods are possible:

- ***Zig zag scan:*** The coefficients are read out diagonally.
- ***Alternate-horizontal scan:*** The coefficients are read out with an emphasis on the horizontal direction first.
- ***Alternate-vertical scan:*** Similar to the horizontal scan, but applied in the vertical direction.

The type of DC prediction determines type of scan that is to be performed. If there is no DC prediction, the zig zag scan is used; if there is DC prediction in horizontal direction, alternate-vertical scan is used, if DC prediction is performed from the vertical direction, the alternate-horizontal scan is used.
The run length encoding can use two different VLC tables, using the value of the quantizer to determine which VLC table is used.

5.6 Interlaced Coding

When the video content to be coded is interlaced, additional coding efficiency can be achieved by adaptively switching between field and frame coding. Texture coding can be performed in field DCT mode or frame DCT mode, switchable on a macroblock basis, and defined by:

- ***Frame DCT coding:*** Each luminance block is composed of lines from two fields alternately.
- ***Field DCT coding:*** Each luminance block is composed of lines from only one of the two fields.

Figure 9 illustrates an example of frame and field DCT coding. The field DCT mode applies to luminance blocks only as chrominance blocks are always coded in frame DCT mode.

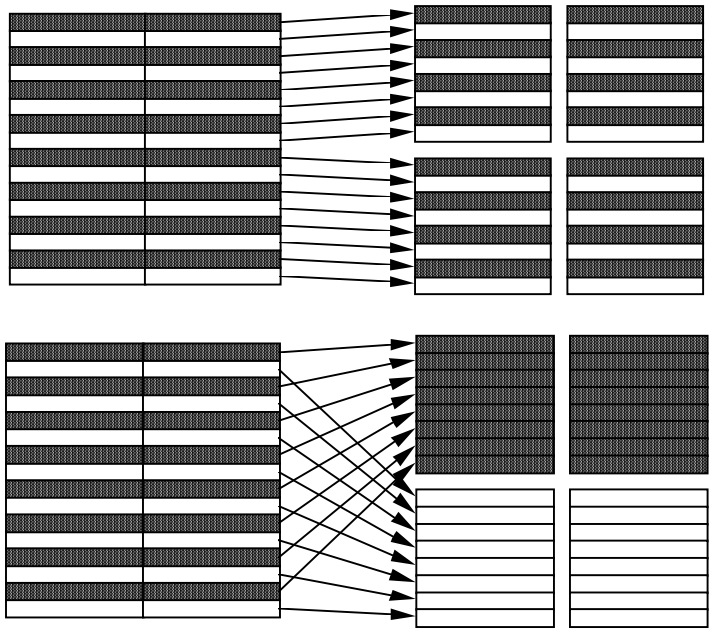


Figure 9: Luminance macroblock structure in frame DCT coding (above), and field DCT coding (below)

5.7 Static texture

One of the functionalities supported by MPEG-4 is the mapping of static textures onto 2-D or 3-D surfaces. MPEG-4 visualsupports this functionality by providing a separate mode for encoding static texture information. The static texture coding technique provides a high degree of scalability, more than the DCT based texture coding technique. The static coding technique is based on a wavelet transform, where the AC and DC bands are coded separately. The wavelet coefficients are quantized, and encoded using a zero-tree algorithm and arithmetic coding.

5.7.1 Wavelet

Texture information is separated into subbands by applying a discrete wavelet transform to the data. The inverse discrete wavelet transform is applied on the subbands to synthesize the texture information from the bitstream. The discrete wavelet transform can either be applied in floating point, or in integer, which is signaled in the bitstream.

The discrete wavelet transform is applied recursively on the obtained subbands, yielding a decomposition tree of subbands. An example of a wavelet transform with two decomposition levels, is shown in Fig. 10. The original texture is decomposed into four subbands, and the lowest frequency subband is split again into four subbands. Here, subband 1 represents the lowest spectral band, and is called the DC component. The other subbands are called the AC subbands. DC and AC subbands are processed differently.

5.7.2 DC subband

The wavelet coefficients of the DC subband are treated differently from the other subbands. The coefficients are coded using a predictive scheme. Each coefficient can be predicted from its left or its top neighbor. The choice of the predictor coefficient depends on the magnitude of the horizontal and vertical gradient of the neighboring coefficients. If the horizontal gradient is smallest, then prediction from the left neighboring coefficient is performed, otherwise prediction from the top neighboring coefficient is performed. The coefficient is then quantized, and encoded using arithmetic coding.

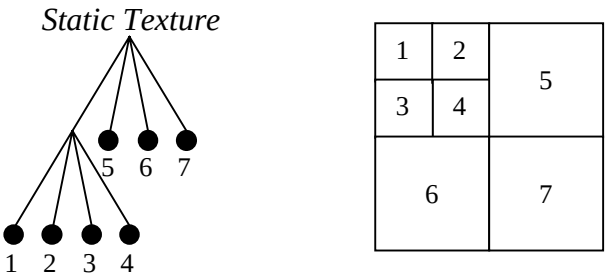


Figure 10: Illustration of a wavelet transform with two decomposition levels.

5.7.3 AC subbands

The wavelet coefficients in the remaining subbands are processed in the following way. Typically many coefficients in the remaining subbands become zero after quantization, and the coding efficiency depends heavily on encoding both the value and the location of the non-zero coefficients effectively. The technique used to achieve a high efficiency is based on the strong correlation between the amplitudes of the wavelet coefficients across scales, at the same spatial location, and of similar orientation. Thus, the coefficient at a coarse scale, the parent, and its descending coefficients at a finer scale, the children, exhibit a strong correlation. These relationships are illustrated in Fig. 11.

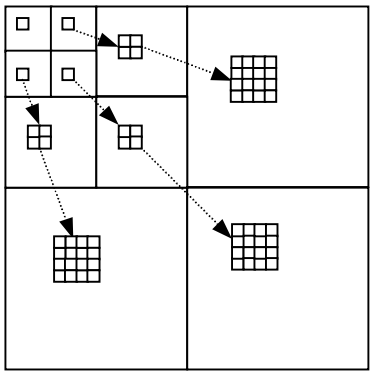


Figure 11: Parent child relationship of wavelet coefficients

Zero tree algorithm exploiting such relationships is used to code both coefficients values and locations. The algorithm relies on the fact that if a wavelet coefficient is zero at a coarse scale, it is very likely that its descendent coefficients are also zero, forming a tree of zeros. Zero trees exist at any tree node where the coefficient is zero, and all its

descendants are also zero. Using this principle, wavelet coefficients in the tree are encoded by arithmetic coding, using a symbol that indicates if a zero tree exists, and the value of the coefficient.

6 Error resilience

This functionality is important for universal access through error-prone environments, such as mobile communications.
MPEG-4 provides several mechanisms to allow error resilience with different degrees of robustness and complexities [4]. These mechanisms are offered by tools providing means for resynchronization, error detection, data recovery and error concealment. There are four error resilience tools in MPEG-4 visual, namely, resynchronization, data partitioning, header extension code, and reversible variable length codes.

- **Resynchronization**: This is the most frequent way to bring error resilience to a bitstream. It consists of inserting unique markers in the bitstream so that in the case of an error, the decoder can skip the remaining bits until the next marker and restart decoding from that point on. MPEG-4 allows for insertion of resynchronization markers after an approximately constant number of coded bits (video packets), as opposed to MPEG-2 and H.263 which allow for resynchronization after a constant number of coded macroblocks (typically a row of macroblocks). Experiments show that the former is a more efficient way of recovering from transmission errors.
- **Data partitioning**: This method separates the bits for coding of motion information and those for the texture information. In the event of an error, a more efficient error concealment may be applied when for instance the error occurs on the texture bits only, by making use of the decoded motion information.
- **Header extension code**: These binary codes allow an optional inclusion of redundant header information, vital for correct decoding of video. This way, the chances of corruption of header information and complete skipping of large portions of bitstream will be reduced.
- **Reversible VLCs:** These VLCs allow to further reduce the influence of error occurrence on the decoded data. RVLCs are codewords which can be decoded in forward as well as backward manners. In the event of an error and skipping of the bitstream until the next resynchronization marker, it is possible to still decode portions of the corrupted bitstream in the reverse order to limit the influence of the error.

Figure 12 summarizes the influence of these tools on the MPEG-4 bitstream syntax.

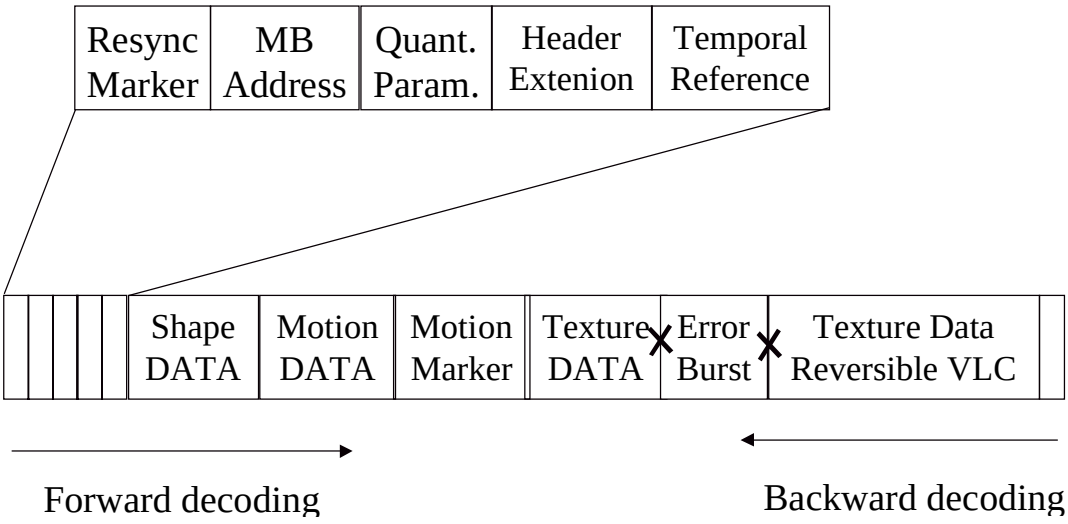


Figure 12: Error resilience tools in MPEG-4

7 Sprite coding

A *sprite* consists of those regions of a VO that are present in the scene, throughout the video segment. An obvious example is a `background sprite' (also referred to as the `background mosaic'), which would consist of all pixels belonging to the background in a camera-panning sequence. This is essentially a static image that could be transmitted only once, at the beginning of the transmission. Sprites have been included in MPEG-4 mainly because they provide high compression efficiency in such cases. For any given instant of time, the background VOP can be extracted by warping/cropping this sprite appropriately. Sprite-based coding is very suitable for synthetic objects, but can also be used for objects in natural scenes that undergo rigid motion.

Similar to the representation of VOPs, the texture information for a sprite is represented by one luminance component and two chrominance components. The three components are processed separately, but the methods used for processing the chrominance components are the same as those used for the luminance components, after appropriate scaling. Shape and texture information for a sprite is encoded as for an I-VOP.

Static sprites are generated, before the encoding process begins, using the original VOPs. The decoder receives each static sprite before the rest of the video segment. The static sprites are encoded in such a way that the reconstructed VOPs can be generated easily, by warping the quantized sprite with the appropriate parameters.
In order to support low-latency applications, several possibilities are envisaged for the transmission of sprites. One way to meet the latency requirements is to transmit only a portion of the sprite in the beginning. The transmitted portion should be sufficient for reconstructing the first few VOPs. The remainder of the sprite is transmitted, piece-wise,

as required or as the bandwidth allows. Another method is to transmit the entire sprite in a progressive fashion, starting with a low quality version, and gradually improving its quality by transmitting residual images. In practice, a combination of these methods can be used.

8 Scalability

Spatial scalability and temporal scalability are both implemented using multiple VOLs. Consider the case of two VOLs: the *base-layer* and the *enhancement-layer*. For spatial scalability, the enhancement-layer improves upon the spatial resolution of a VOP provided by the base-layer. Similarly, in the case of temporal scalability, the enhancement-layer may be decoded if the desired frame rate is higher than that offered by the base-layer. Thus, temporal scalability improves the smoothness of motion in the sequence. Object scalability is naturally supported by the VO based scheme. At present, only rectangular VOPs are supported in MPEG-4, as far as spatial scalability is concerned.

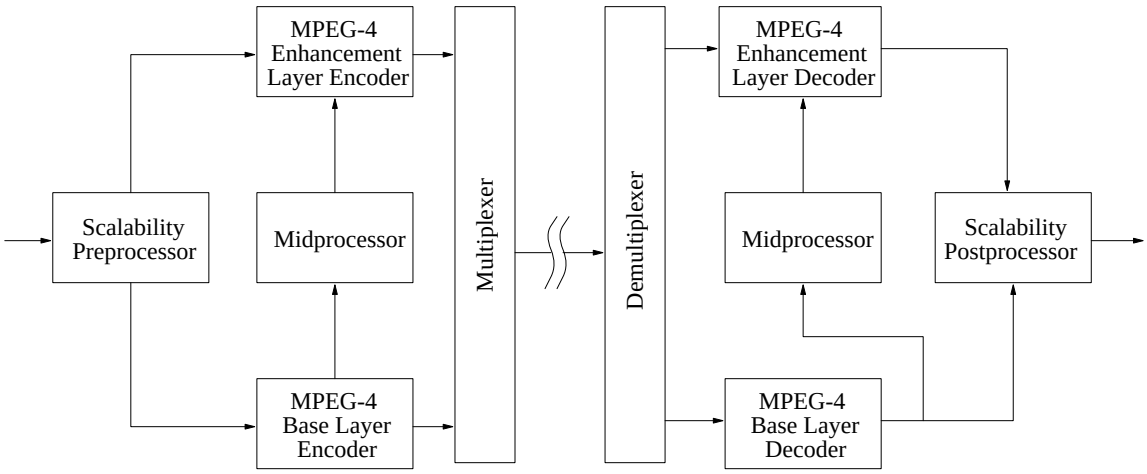


Figure 13: Block diagram of MPEG-4 generalized scalability framework. The specific algorithms implemented in the preprocessor, midprocessor and postprocessor depend upon the type of scalability being enabled.

MPEG-4 uses a *generalized scalability* framework to enable spatial and temporal scalabilities. This framework allows the inclusion of separate modules, as necessary, to enable the various scalabilities. As shown in Fig. 13, a scalability preprocessor is used to implement the desired scalability. It operates on VOPs. For example, in the case of spatial scalability, the preprocessor down-samples the input VOPs to produce the base-layer VOPs that are processed by the VOP encoder. The *midprocessor* takes the reconstructed base-layer VOPs and up-samples them. The difference between the original VOP and the output of the midprocessor forms the input to the encoder for the enhancement layer. To implement temporal scalability, the preprocessor separates out the

frames into two streams. One stream forms the input for the base-layer encoder, while the other is processed by the enhancement-layer encoder. The midprocessor is bypassed in this case.

8.1 Spatial Scalability

The base-layer VOPs are encoded in the same way as in the non-scalable case discussed in previous sections. VOPs of the enhancement layer are encoded as P-VOPs or B-VOPs.
If a VOP in the enhancement-layer is temporally coincident with an I-VOP in the base-layer, it could be treated as a P-VOP. VOPs in the enhancement-layer that are coincident with P VOPs in the base-layer could be coded as B-VOPs. Since the base-layer serves as the reference for the enhancement layer, VOPs in the base-layer must be encoded before their corresponding VOPs in the enhancement layer. Figure 14 illustrates an example of how the enhancement layer can be decoded from the base layerusing spatial scalability.

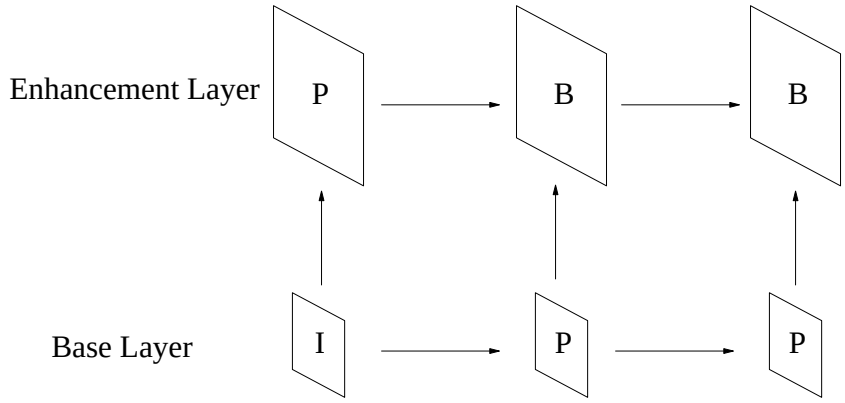


Figure 14: Illustration of base and enhancement layer behavior in the case of spatial scalability.

8.2 Temporal Scalability

In temporal scalability, the frame rate of the visual data is enhanced. Enhancement-layers carry information to be visualized between the frames of the base-layer. The enhancement layer may act in one of two ways, as shown in Fig. 15:

- ***Type I:*** The enhancement-layer improves the resolution of only a portion of the base-layer.
- ***Type II:*** The enhancement-layer improves the resolution of the entire base-layer.

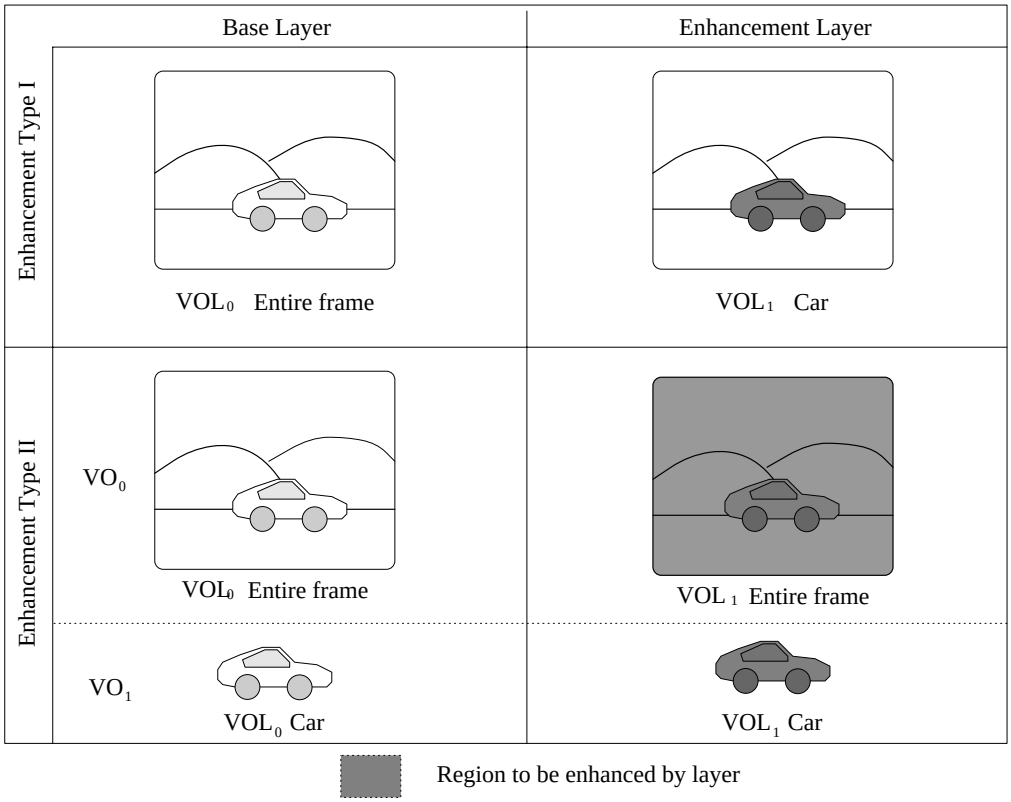


Figure 15: The two enhancement types in MPEG-4 temporal scalability. In enhancement type I, only a selected region of the VOP (i.e. just the car) is enhanced, while the rest (i.e. the landscape) is not. In enhancement type II, enhancement is applicable only at entire VOP level.

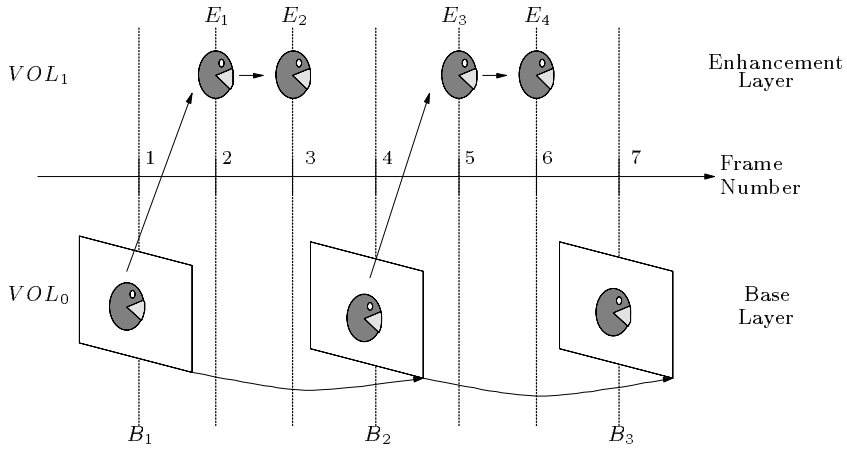


Figure 16: Base and enhancement layer behavior for temporal scalability for type I enhancement (improving only a portion of the base-layer).

An example of type I enhancement is shown in Fig. 16, whereas Fig. 17 illustrates an example of type II enhancement.

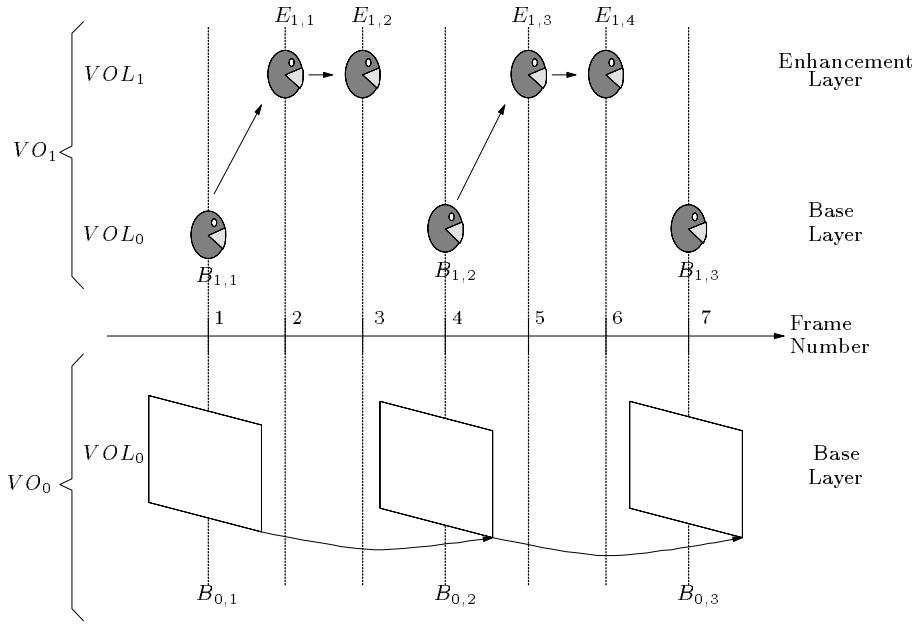


Figure 17: Base and enhancement layer behavior for temporal scalability for type II enhancement (improving the entire base-layer).}

9 Conformance points

Conformance points form the basis for interoperability, the main driving force behind standardization. Implementations of all manufacturers that conform to a particular conformance point are interoperable with each other. Conformance tests can be carried out to test and to show conformance, and thus promoting interoperability. For this purpose bitstreams should be produced for a particular conformance point, that can test decoders that are conforming to that conformance point.
MPEG-4 defines conformance points in the form of profiles and levels. Profiles and levels define subsets of the syntax and semantics of MPEG-4, which in turn define the required decoder capabilities. An MPEG-4 natural video profile is a defined subset of the MPEG-4 video syntax and semantics, specified in the form of a set of tools, or as object types. Object types group together MPEG-4 video tools that provide a certain

functionality. A level within a profile defines constraints on parameters in the bitstream and their corresponding tools.

9.1 Object types

An object type defines a subset of tools of the MPEG-4 visual standard that provides a functionality, or a group of functionalities. Object types are the building blocks of MPEG-4 profiles, and as such simplify the definition of profiles. There are 6 natural video object types, namely, simple, core, main, simple scalable, N-bit, and still scalable texture. In Tab. 1 the video object types are described.

9.2 Profiles

MPEG-4 video profiles are defined in terms of video object types. The 6 video profiles are described in Tab. 2.

9.3 Levels

A level within a profile defines constraints on parameters in the bitstream that relate to the tools of that profile. Currently there are 11 natural video profile and level definitions that each constrains about 15 parameters that are defined for each level. To provide some insight into the levels, for the three most important profiles, core, simple, and main, a subset of level constraints is given in Tab. 3. The macroblock memory size is the bound on the memory (in macroblock units) which can be used by the (Video reference Memory Verifier) VMV algorithm. This algorithm models the pixel memory needed by the entire visual decoding process.

MPEG-4 video tools	MPEG-4 video object types					
	Simple	Core	Main	Simple Scalable	N-bit	Still Scalable Texture
Basic(I and P-VOP, coefficient prediction, 4-MV, unrestricted MV)	●	●	●	●	●	
Error resilience	●	●	●	●	●	
Short Header	●	●	●		●	
B-VOP		●	●	●	●	
P-VOP with OBMC (Texture)						
Method 1/Method 2 Quantization		●	●		●	
P-VOP based temporal scalability		●	●		●	
Binary Shape		●	●		●	
Grey Shape			●			
Interlace			●			
Sprite			●			
Temporal Scalability (Rectangular)				●		
Spatial Scalability (Rectangular)				●		
N-Bit					●	
Scalable Still Texture						●

Table 1: MPEG-4 video object types

MPEG-4 video object types	MPEG-4 video profiles					
	Simple	Core	Main	Simple Scalable	N-Bit	Scalable Texture
Simple	●	●	●	●	●	
Core		●	●		●	
Main			●			
Simple Scaleable				●		
N-Bit					●	
Scalable Still Texture			●			●

Table 2: MPEG-4 video profiles

Profile and Level		Typical scene size	Bitrate (bit/sec)	Maximum number of objects	Total mblk memory (mblk units)
Simple Profile	L1	QCIF	64 k	4	198
	L2	CIF	128 k	4	792
	L3	CIF	384 k	4	792
Core Profile	L1	QCIF	384 k	4	594
	L2	CIF	2 M	16	2376
Main Profile	L2	CIF	2 M	16	2376
	L3	ITU-R 601	15 M	32	9720
	L4	1920x1088	38.4 M	32	48960

Table 3: Subset of MPEG-4 video profile and level definitions

9.4 Rate control

An important conformance point is the maximum bitrate, or the maximum size of the decoder bitstream buffer. Therefore, while encoding a video scene, rate control and buffer regulation algorithms are important building blocks in the implementation of the MPEG-4 video standard. In a variable bitrate (VBR) environment, the rate control scheme attempts to achieve optimum quality for the decoded scene given a target bitrate. In constant bitrate (CBR) applications, the rate controller has to meet the constraints of fixed latency and buffer size. To meet these requirements, the rate control algorithm

controls the quantization parameters. As a guideline to implementers, the MPEG-4 video standard describes a possible implementation of a rate control algorithm. The algorithm uses a Scalable Rate Control scheme (SRC) that can satisfy both VBR and CBR requirements. The SRC is based on the assumption that rate-distortion function can be modeled by the following equation:

$$R = \frac{X_1 S}{Q} + \frac{X_2 S}{Q^2}$$

where R is the rate, X_1 and X_2 are modeling parameters, S is a measure of activity in the frame, and Q is the quantization parameter. The first and second order coefficients, X_1 and X_2, are initialized at the beginning of the process and updated based on the encoding results of each frame. The quantization parameter Q is computed based on this equation.
This scheme achieves frame rate control for both CBR and VBR cases. A rate control algorithm for multiple video objects is derived from this algorithm, by using a bit allocation table based on Human Visual Sensitivity (HVS) of color tolerance. This results in a bit allocation for each object. Next, the number of coded bits per block are estimated based on block variance classification, and a bits estimation model. This results in a reference quantization parameter. The object is encoded using this reference parameter. While encoding the object, small adjustments to the reference parameter can be made depending on the possible deviation of the predicted bits from the actual bits. If the number of actual bits produced is much higher than the allocated bit budget, a frame skip parameter is computed that allows to skip several instances of the video object to reduce the current buffer level.

9.5 Video verifier models

In order to build competitive products with economical implementations, the MPEG-4 video standard provides models that provides bounds on memory and computational requirements. To this purpose a video verifier is defined, consisting of three normative models, the video rate buffer verifier, the video complexity verifier, and the video reference memory verifier. These models are used in the definition of the conformance points, the profiles and levels described above.

- ***Video rate buffer verifier (VBV):*** The VBV provides bounds on the memory requirements of the bitstream buffer needed by a video decoder. Conforming video bitstreams can be decoded with a predetermined buffer memory size. When an MPEG-4 video bitstream consists of more then one video object, the VBV is applied to each VOL independently. In the VBV model the encoder models a virtual buffer, where bits arrive at the rate produced by the encoder, and bits are removed instantaneously at decoding time. The size of the VBV is specified in the profile and level indication. A VBV conformant bitstream is produced if this virtual buffer never underflows or overflows. Assuming a constant delay, a hypothetical decoder would

need a buffer size of the size of the VBV buffer with a guarantee that the decoder buffer never underflows or overflows.

- ***Video complexity verifier (VCV):*** The VCV provides bounds on the processing speed requirements of a video decoder. Conformant bitstreams can be decoded by a video processor with predetermined processor capability in terms of the number of macroblocks per second it can process. The VCV model is applied to all macroblocks in an MPEG-4 video bitstream. In the VCV a virtual macroblock buffer accumulates all macroblocks encoded by the encoder, and added instantaneously to the buffer. The buffer has a pre-specified macroblock capacity, as specified by profile and level indication. In addition, the minimum rate by which macroblocks are decoded is specified as well in the profile and level indication. With these two parameters specified, an encoder can compute how many macroblocks it can produce at any time instant to produce a VCV conformant bitstream. Such a bitstream has the guarantee that a conformant decoder with the specified processor capability to decode all macroblocks.

- ***Video reference memory verifier (VMV):*** The VMV provides bounds on the macroblock (pixel) memory requirements of a video decoder. Conformant bitstreams can be decoded by a video processor with predetermined pixel memory size. The VMV is applied to all macroblocks in an MPEG-4 video bitstream. The hypothetical decoder fills the VMV buffer at the same macroblock decoding rate as the VCV model. The amount of reference memory needed to decode a VOP is defined as the total number of macroblocks in a VOP. For reference VOPs (I or P) the total memory allocated to the previous reference VOP is released at presentation time plus the VCV latency, whereas for B-VOPs the total memory allocated to that B-VOP is released at presentation time plus the VCV latency. A VMV conformant bitstream can be guaranteed decoded by a conformant decoder that has the specified amount of macroblock memory.

For an encoder to produce a bitstream compliant with a certain profile and level specification, it has to simulate the VBV, VCV, and VMV together. In Fig. 18, the complete video verifier model, and the relations between the three video verifiers is shown.

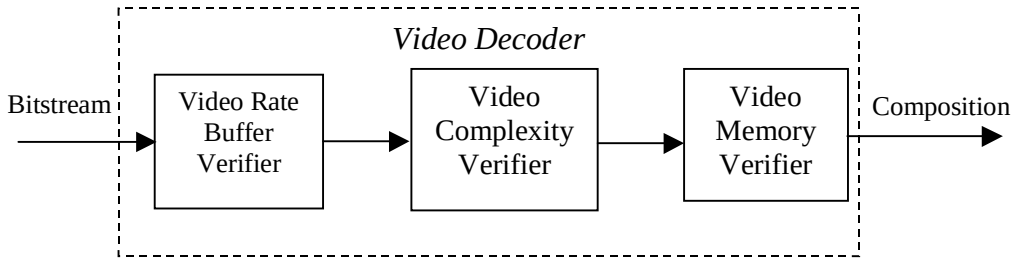


Figure 18: Video verifier model

10 Concluding remarks

MPEG-4 has been developed to support a wide range of multimedia applications. Past standards mainly concentrated on deriving as compact a representation as possible, of the video (and associated audio) data, whereas in order to support the various applications envisaged, MPEG-4 enables functionalities that are required by many such applications

MPEG-4 visual standard uses an object-based representation of the video sequence at hand. This allows easy access and manipulation of arbitrarily shaped regions in frames of the video. The structure based on Video Objects (VOs) directly supports one highly desirable functionality: object-based interactivity. Spatial scalability and temporal scalability are also supported in MPEG-4. Scalability is implemented in terms of layers of information, where the minimum needed to decode is the base-layer. Any additional enhancement-layer will improve the resulting image quality either in temporal or in spatial resolution. Sprite and image texture coding are two new features supported by MPEG-4.

To accommodate universal access, transmission oriented functionalities have also been considered in MPEG-4. Functionalities for error robustness and error resilience handle transmission errors, and the rate control functionality adapts the encoder to the available channel bandwidth.

New tools are still being added to the MPEG-4 coding algorithm in the form of an amendment to the standard. Extensive tests show that MPEG-4 achieves better or similar image qualities at all bitrates targeted, with the bonus of added functionalities.

References

[1] MPEG-1 Video Group, "Information Technology - Coding of Moving Pictures and Associated Audio for Digital Storage Media up to about 1.5 Mbit/s: Part 2 - Video," ISO/IEC 11172-2, International Standard, 1993.

[2] MPEG-2 Video Group, "Information Technology - Generic Coding of Moving Pictures and Associated Audio: Part 2 - Video," ISO/IEC 13818-2, International Standard, 1995.

[3] L. Chariglione, “MPEG and Multimedia Communications”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 7, No. 1, pp. 5-18, Feb. 1997.

[4] MPEG-4 Video Group, “Generic Coding of Audio-Visual Objects: Part 2 - Visual,” ISO/IEC JTC1/SC29/WG11 N1902, FDIS of ISO/IEC 14496-2, Atlantic City, Nov. 1998

[5] ITU-T Experts Group on Very Bitrate Visual Telephony, “ITU-T Recommendation H.263: Video Coding for Low Bitrate Communication,” Dec. 1995.

[6] C. Le Buhan, F. Bossen, S. Bhattacharjee, F. Jordan, T. Ebrahimi, “Shape representation and coding of visual objects in multimedia applications - An Overview”, (invited paper), in Annales des Telecommunications 53, No 5-6, pp. 164-178, May 1998.

[7] F. Bossen, T. Ebrahimi, "A simple and efficient binary shape coding technique based on bitmap representation" in Proc. of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP'97), vol. 4, pp. 3129-3132, Munich, Germany, April 20-24, 1997.

Biographies

Touradj Ebrahimi was born on July 30, 1965. He received his M.Sc. and Ph.D., both in Electrical Engineering, from the Swiss Federal Institute of Technology, Lausanne, Switzerland, in 1989 and 1992 respectively. From 1989 to 1992, he was a research assistant at the Signal Processing Laboratory of the Swiss Federal Institute of Technology (EPFL). During the summer 1990, he was a visiting researcher at the Signal and Image Processing Institute of the University of Southern California, Los Angeles, California. In 1993, he was a research engineer at the Corporate Research Laboratoriess of Sony Corporation in Tokyo, where he conducted research on advanced video compression techniques for storage applications. In 1994, he served as a research consultant at AT&T Bell Laboratories working on very low bitrate video coding. He is currently a Professor at the Signal Processing Laboratory of EPFL, where he is involved with various aspects of Digital Video and Multimedia applications and in charge of the Digital TV group. In 1989, he was the recipient of the IEEE and Swiss national ASE award. His research

interests are multidimensional signal processing, image processing, information theory, and coding. He is the author or the co-author of more than 70 research publications, and 6 patents. He is a member of the IEEE, SPIE and EURASIP.

Caspar Horne received the M.Sc. degree from the Technical University of Delft, the Netherlands, in 1986, and the Ph.D. degree from the Ecole Polytechnique Féderale de Lausanne, Switzerland, in 1990, both in electrical engineering
From 1986 to 1987 he was associated with the Ecole Polytechnique Féderale de Lausanne, where he worked, in cooperation with Hoffmann Laroche, Basel, Switzerland, on ultra-sonic blood flow mapping of high velocity blood streams to visualize anomalies in the heart. From 1987 to 1990 he worked on unsupervised image segmentation using multi-dimensional texture feature sets for computer vision applications, in cooperation with Thomson CSF, Rennes, France. From 1991 to 1993 he was post-doctoral member of technical staff with the visual communications research department of AT&T Bell Laboratories in Holmdel, New Jersey, working on video coding, MPEG2 standards development, and MPEG1 and H.261 hardware implementations. From 1993 to 1995 he was with the research laboratories and the video and networking division of Tektronix, Inc. Beaverton, Oregon, as research scientist, working on video coding for studio quality applications, resulting in the MPEG2 4:2:2 Profile, and was actively involved in the MPEG-4 standard development. He is currently with Mediamatics Inc. in Fremont, California involved in developing cost-effective multi-media solutions based on MPEG standards, and industry standards such as DVD, both in software and hardware, for the PC market and for the consumer electronics market. In addition he is actively involved in the MPEG standards development, and participating in the ATSC and DAVIC standards developments. Dr. Horne has been actively involved in MPEG standards development since 1991. He was the first editor of the MPEG-4 SNHC Verification Model, and the editor-in-chief of part 2 (visual) of the MPEG-4 standard. He has been awarded three patents in the area of video coding, and has published over 30 refereed technical papers in the areas of image processing, and image and video coding, and several book chapters in the areas of image processing and video coding. He received a Bell Laboratories Merit Award in 1991 for work on video standards implementations, and an ISO award for outstanding contributions to the MPEG-4 standard in 1998.