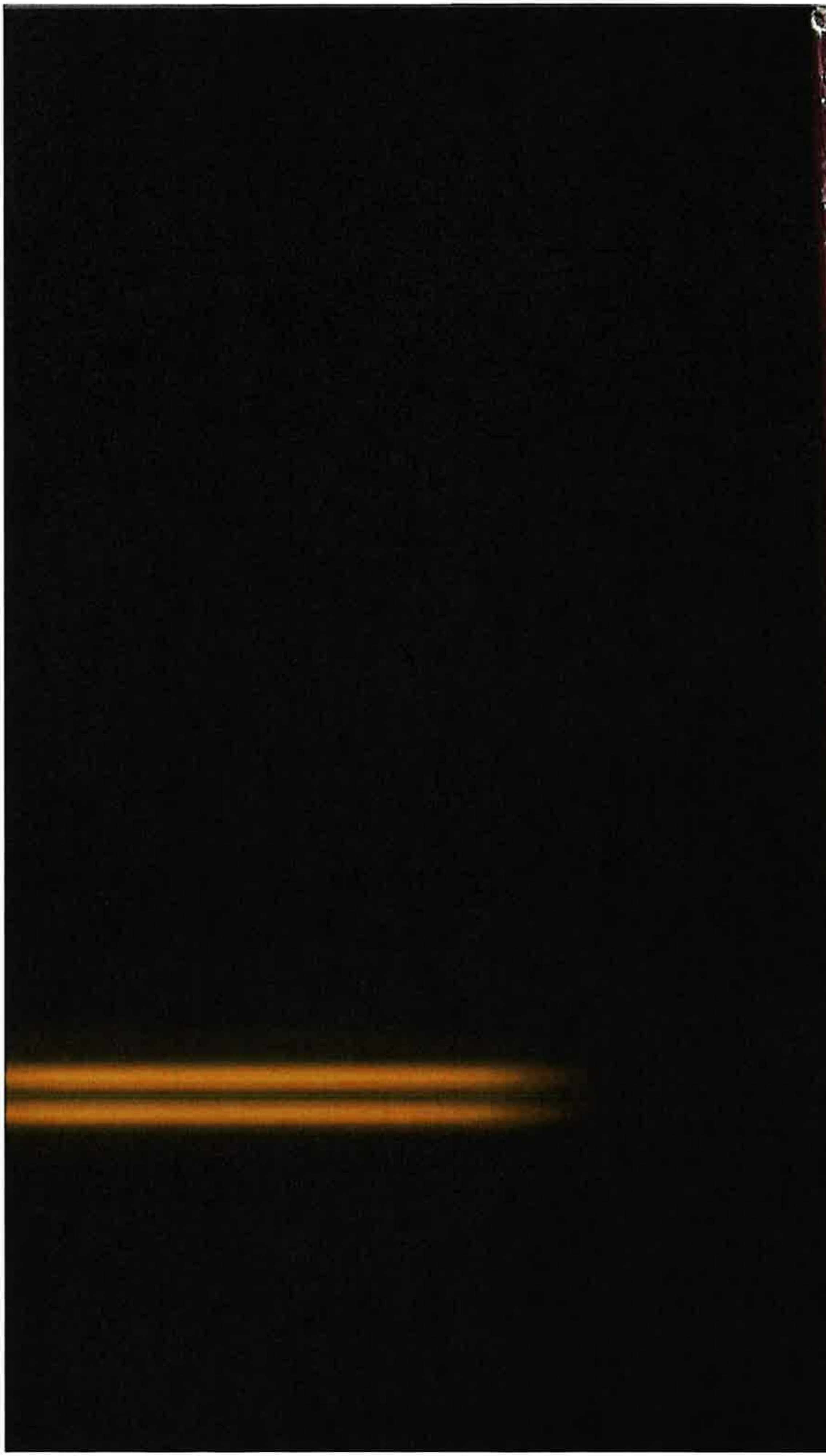
Handbook of Image & Video Processing

EDITOR AL BOVIK



BOVIK

Handbook of Image & Video Processing

AP

Academic Press

Academic Press Series in Communications, Networking, and Multimedia

EDITOR-IN-CHIEF

Jerry D. Gibson
Southern Methodist University

This series has been established to bring together a variety of publications that represent the latest in cutting-edge research, theory, and applications of modern communication systems. All traditional and modern aspects of communications as well as all methods of computer communications are to be included. The series will include professional handbooks, books on communication methods and standards, and research books for engineers and managers in the world-wide communications industry.

This book is printed on acid-free paper. ♾

ACADEMIC PRESS
A Harcourt Science and Technology Company
525 B Street, Suite 1900, San Diego, CA 92101-4495, USA
http://www.academicpress.com

Academic Press
Harcourt Place, 32 Jamestown Road, London, NW1 7BY, UK
http://www.hbuk.co.uk/ap/

Library of Congress Catalog Number: 99-69120

ISBN: 0-12-119790-5

Printed in Canada

00 01 02 03 04 05 FR 9 8 7 6 5 4 3 2 1

6.1

Basic Concepts and Techniques of Video Coding and the H.261 Standard

Barry Barnett
The University of Texas
at Austin

1 Introduction

The subject of video coding is of fundamental importance to many areas in engineering and the sciences. Video engineering is quickly becoming a largely digital discipline. The digital transmission of television signals via satellites is commonplace, and widespread HDTV terrestrial transmission is slated to begin in 1999. Video compression is an absolute requirement for the growth and success of the low-bandwidth transmission of digital video signals. Video encoding is being used wherever digital video communications, storage, processing, acquisition, and reproduction occur. The transmission of high-quality multimedia information over high-speed computer networks is a central problem in the design of *Quality of Services* (QOS) for digital transmission providers. The *Motion Pictures Expert Group* (MPEG) has already finalized two video coding standards, MPEG-1 and MPEG-2, that define methods for the transmission of digital video information for multimedia and television formats. MPEG-4 is currently addressing the transmission of very low bitrate video. MPEG-7 is addressing the standardization of video storage and retrieval services (Chapters 9.1 and 9.2 discuss video storage and retrieval). A central aspect to each of the MPEG standards are the video encoding and decoding algorithms that make digital video applications practical. The MPEG Standards are discussed in Chapters 6.4 and 6.5.

Video compression not only reduces the storage requirements or transmission bandwidth of digital video applications, but it also affects many system performance tradeoffs. The design and selection of a video encoder therefore is not only based on its ability to compress information. Issues such as bitrate versus distortion criteria, algorithm complexity, transmission channel characteristics, algorithm symmetry versus asymmetry, video source statistics, fixed versus variable rate coding, and standards compatibility should be considered in order to make good encoder design decisions.

The growth of digital video applications and technology in the past few years has been explosive, and video compression is playing a central role in this success. Yet, the video coding discipline is relatively young and certainly will evolve and change significantly over the next few years. Research in video coding has great vitality and the body of work is significant. It is apparent that this relevant and important topic will have an immense affect on the future of digital video technologies.

2 Introduction to Video Compression

Video or visual communications require significant amounts of information transmission. Video compression, as considered here, involves the bitrate reduction of a digital video signal carrying visual information. Traditional video-based compression, like other information compression techniques, focuses on eliminating the redundant elements of the signal. The degree to which the encoder reduces the bitrate is called its *coding efficiency*; equivalently, its inverse is termed the *compression ratio*:

$$\begin{aligned}\text{coding efficiency} &= (\text{compression ratio})^{-1}\\ &= \text{encoded bitrate/decoded bitrate.} \quad (1)\end{aligned}$$

Compression can be a lossless or lossy operation. Because of the immense volume of video information, lossy operations are mainly used for video compression. The loss of information or distortion measure is usually evaluated with the *mean square error* (MSE), *mean absolute error* (MAE) criteria, or *peak signal-to-noise ratio* (PSNR):

$$\text{MSE} = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}[I(i,j) - \hat{I}(i,j)]^2$$

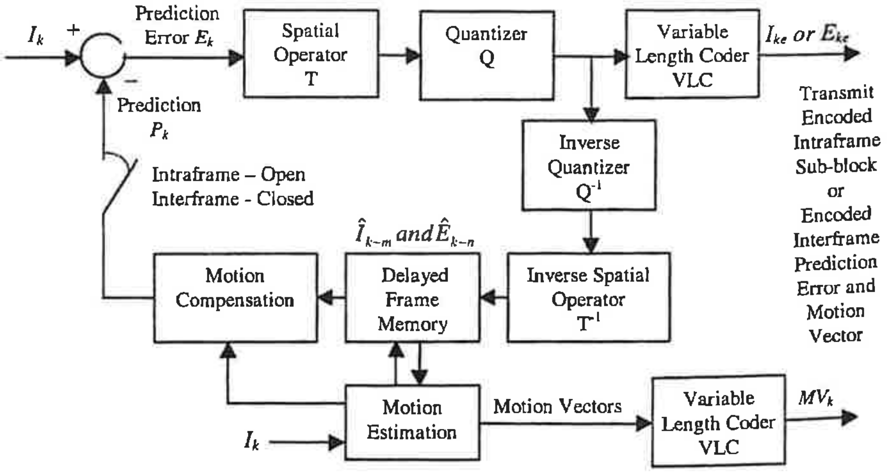
$$\text{MAE} = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}|I(i,j) - \hat{I}(i,j)|$$

$$\text{PSNR} = 20\log_{10}\left(\frac{2^n}{\text{MSE}^{1/2}}\right), \quad (2)$$

for an image I and its reconstructed image $\hat{I}$, with pixel indices $1 \leq i \leq M$ and $1 \leq j \leq N$, image size $N \times M$ pixels, and n bits per pixel. The MSE, MAE, and PSNR as described here are global measures and do not necessarily give a good indication of the reconstructed image quality. In the final analysis, the human observer determines the quality of the reconstructed image and video quality. The concept of distortion versus coding efficiency is one of the most fundamental tradeoffs in the technical evaluation of video encoders. The topic of perceptual quality assessment of compressed images and video is discussed in Section 8.2.

Video signals contain information in three dimensions. These dimensions are modeled as *spatial* and *temporal* domains for video encoding. Digital video compression methods seek to minimize information redundancy independently in each domain. The major international video compression standards (MPEG-1, MPEG-2, H.261) use this approach. Figure 1 schematically depicts a generalized video compression system that implements the spatial and temporal encoding of a digital image sequence. Each image in the sequence I_k is defined as in Eq. (1). The spatial encoder operates on image *blocks*, typically of the order of 8 × 8 pixels each. The temporal encoder generally operates on 16 × 16 pixel image blocks. The system is designed for two modes of operation, the *intraframe* mode and the *interframe* mode.

The single-layer feedback structure of this generalized model is representative of the encoders that are recommended by the *International Standards Organization* (ISO) and *International Telecommunications Union* (ITU) video coding standards, MPEG-1, MPEG-2/H.262, and H.261 [1–3]. The feedback loop is used in the interframe mode of operation and generates a *prediction error* between the blocks of the current frame and the current prediction frame. The prediction is generated by the *motion compensator*. The *motion estimation* unit creates *motion vectors* for each 16 × 16 block. The motion vectors and previously reconstructed frame are fed to the motion compensator to create the prediction.



I_k
+
−
Prediction Error E_k
Spatial Operator T
Quantizer Q
Variable Length Coder VLC
I_{ke} or E_{ke}
Transmit Encoded Intraframe Sub-block or Encoded Interframe Prediction Error and Motion Vector
Prediction P_k
Intraframe – Open
Interframe - Closed
Inverse Quantizer Q^{-1}
$\hat{I}_{k-m}$ and $\hat{E}_{k-n}$
Motion Compensation
Delayed Frame Memory
Inverse Spatial Operator T^{-1}
Motion Estimation
Motion Vectors
Variable Length Coder VLC
MV_k
I_k

FIGURE 1 Generalized video compression system.

The intraframe mode spatially encodes an entire current frame on a periodic basis, e.g., every 15 frames, to ensure that systematic errors do not continuously propagate. The intraframe mode will also be used to spatially encode a block whenever the interframe encoding mode cannot meet its performance threshold. The intraframe versus interframe mode selection algorithm is not included in this diagram. It is responsible for controlling the selection of the encoding functions, data flows, and output data streams for each mode.

The Intraframe encoding mode does not receive any input from the feedback loop. I_k is spatially encoded, and losslessly encoded by the variable length coder (VLC) forming I_{ke}, which is transmitted to the decoder. The receiver decodes I_{ke}, producing the reconstructed image subblock $\hat{I}_k$. During the interframe coding mode, the current frame prediction P_k is subtracted from the current frame input I_k to form the current prediction error E_k. The prediction error is then spatially and VLC encoded to form E_{ke}, and it is transmitted along with the VLC encoded motion vectors MV_k. The decoder can reconstruct the current frame $\hat{I}_k$ by using the previously reconstructed frame $\hat{I}_{k-1}$ (stored in the decoder), the current frame motion vectors, and the prediction error. The motions vectors MV_k operate on $\hat{I}_{k-1}$ to generate the current prediction frame P_k. The encoded prediction error $\hat{E}_{ke}$ is decoded to produce the reconstructed prediction error $\hat{E}_k$. The prediction error is added to the prediction to form the current frame $\hat{I}_k$. The functional elements of the generalized model are described here in detail.

1. Spatial operator: this element is generally a unitary two-dimensional linear transform, but in principle it can be any unitary operator that can distribute most of the signal energy into a small number of coefficients, i.e., decorrelate the signal data. Spatial transformations are successively applied to small image blocks in order to take advantage of the high degree of data correlation in adjacent image pixels. The most widely used spatial operator for image and video coding is the *discrete cosine transform* (DCT). It is applied to 8 × 8 pixel image blocks and is well suited for image transformations because it uses real computations with fast implementations, provides excellent decorrelation of signal components, and avoids generation of spurious components between the edges of adjacent image blocks.
2. Quantizer: the spatial or transform operator is applied to the input in order to arrange the signal into a more suitable format for subsequent lossy and lossless coding operations. The quantizer operates on the transform generated coefficients. This is a lossy operation that can result in a significant reduction in the bitrate. The quantization method used in this kind of video encoder is usually scalar and non-uniform. The scalar quantizer simplifies the complexity of the operation as compared to *vector quantization* (VQ). The non-uniform quantization interval is sized according to the distribution of the transform coefficients in order to minimize the bitrate and the distortion created by the quantization process. Alternatively, the quantization interval size can be adjusted based on the performance of the *human Visual System* (HVS). The *Joint Pictures Expert Group* (JPEG) standard includes two (luminance and color difference) HVS sensitivity weighted quantization matrices in its "Examples and Guidelines" annex. JPEG coding is discussed in Sections 5.5 and 5.6.
3. Variable length coding: The lossless VLC is used to exploit the "symbolic" redundancy contained in each block of transform coefficients. This step is termed "entropy coding" to designate that the encoder is designed to minimize the source entropy. The VLC is applied to a serial bit stream that is generated by scanning the transform coefficient block. The scanning pattern should be chosen with the objective of maximizing the performance of the VLC. The MPEG encoder for instance, describes a zigzag scanning pattern that is intended to maximize transform zero coefficient run lengths. The H.261 VLC is designed to encode these run lengths by using a variable length *Huffman* code.

The feedback loop sequentially reconstructs the encoded spatial and prediction error frames and stores the results in order to create a current prediction. The elements required to do this are the inverse quantizer, inverse spatial operator, delayed frame memory, motion estimator, and motion compensator.

1. Inverse operators: The inverse operators Q^{-1} and T^{-1} are applied to the encoded current frame I_{ke} or the current prediction error E_{ke} in order to reconstruct and store the frame for the motion estimator and motion compensator to generate the next prediction frame.
2. Delayed frame memory: Both current and previous frames must be available to the motion estimator and motion compensator to generate a prediction frame. The number of previous frames stored in memory can vary based upon the requirements of the encoding algorithm. MPEG-1 defines a B frame that is a bidirectional encoding that requires that motion prediction be performed in both the forward and backward directions. This necessitates storage of multiple frames in memory.
3. Motion estimation: The temporal encoding aspect of this system relies on the assumption that rigid body motion is responsible for the differences between two or more successive frames. The objective of the motion estimator is to estimate the rigid body motion between two frames. The motion estimator operates on all current frame 16 × 16 image blocks and generates the pixel displacement or *motion vector* for each block. The technique used to generate motion vectors is called *block-matching motion estimation* and is discussed further in Section 5.4. The method uses the current frame I_k and the previous reconstructed frame

$\hat{I}_{k-1}$ as input. Each block in the previous frame is assumed to have a displacement that can be found by searching for it in the current frame. The search is usually constrained to be within a reasonable neighborhood so as to minimize the complexity of the operation. Search matching is usually based on a minimum MSE or MAE criterion. When a match is found, the pixel displacement is used to encode the particular block. If a search does not meet a minimum MSE or MAE threshold criterion, the motion compensator will indicate that the current block is to be spatially encoded by using the intraframe mode.

4. Motion compensation: The motion compensator makes use of the current frame motion estimates MV_k and the previously reconstructed frame $\hat{I}_{k-1}$ to generate the current frame prediction P_k. The current frame prediction is constructed by placing the previous frame blocks into the current frame according to the motion estimate pixel displacement. The motion compensator then decides which blocks will be encoded as prediction error blocks using motion vectors and which blocks will only be spatially encoded.

The generalized model does not address some video compression system details such as the bit-stream syntax (which supports different application requirements), or the specifics of the encoding algorithms. These issues are dependent upon the video compression system design.

Alternative video encoding models have also been researched. Three-dimensional (3-D) video information can be compressed directly using VQ or 3-D *wavelet* encoding models. VQ encodes a 3-D block of pixels as a codebook index that denotes its "closest or nearest neighbor" in the minimum squared or absolute error sense. However, the VQ codebook size grows on the order as the number of possible inputs. Searching the codebook space for the nearest neighbor is generally very computationally complex, but structured search techniques can provide good bitrates, quality, and computational performance. *Tree-structured* VQ (TSVQ) [13] reduces the search complexity from codebook size N to log N, with a corresponding loss in average distortion. The simplicity of the VQ decoder (it only requires a table lookup for the transmitted codebook index) and its bitrate-distortion performance make it an attractive alternative for specialized applications. The complexity of the codebook search generally limits the use of VQ in real-time applications. Vector quantizers have also been proposed for interframe, variable bitrate, and subband video compression methods [4].

Three-dimensional wavelet encoding is a topic of recent interest. This video encoding method is based on the *discrete wavelet transform* methods discussed in Section 5.4. The wavelet transform is a relatively new transform that decomposes a signal into a *multiresolution* representation. The multiresolution decomposition makes the wavelet transform an excellent signal analysis tool because signal characteristics can be viewed in a variety of time-frequency scales. The wavelet transform is implemented in practice by the use of multiresolution or *subband* filterbanks [5]. The wavelet filterbank is well suited for video encoding because of its ability to adapt to the multiresolution characteristics of video signals. Wavelet transform encodings are naturally hierarchical in their time-frequency representation and easily adaptable for *progressive transmission* [6]. They have also been shown to possess excellent bitrate-distortion characteristics.

Direct three-dimensional video compression systems suffer from a major drawback for real-time encoding and transmission. In order to encode a sequence of images in one operation, the sequence must be buffered. This introduces a buffering and computational delay that can be very noticeable in the case of interactive video communications.

Video compression techniques treating visual information in accordance with HVS models have recently been introduced. These methods are termed "second-generation or object-based" methods, and attempt to achieve very large compression ratios by imitating the operations of the HVS. The HVS model can also be incorporated into more traditional video compression techniques by reflecting visual perception into various aspects of the coding algorithm. HVS weightings have been designed for the DCT AC coefficients quantizer used in the MPEG encoder. A discussion of these techniques can be found in Chapter 6.3.

Digital video compression is currently enjoying tremendous growth, partially because of the great advances in VLSI, ASIC, and microcomputer technology in the past decade. The real-time nature of video communications necessitates the use of general purpose and specialized high–performance hardware devices. In the near future, advances in design and manufacturing technologies will create hardware devices that will allow greater adaptability, interactivity, and interoperability of video applications. These advances will challenge future video compression technology to support format-free implementations.

3 Video Compression Application Requirements

A wide variety of digital video applications currently exist. They range from simple low-resolution and low-bandwidth applications (multimedia, PicturePhone) to very high-resolution and high-bandwidth (HDTV) demands. This section will present requirements of current and future digital video applications and the demands they place on the video compression system.

As a way to demonstrate the importance of video compression, the transmission of digital video television signals is presented. The bandwidth required by a digital television signal is approximately one-half the number of picture elements (pixels) displayed per second. The analog pixel size in the vertical dimension is the distance between scanning lines, and the horizontal dimension is the distance the scanning spot moves during

½ cycle of the highest video signal transmission frequency. The bandwidth is given by Eq (3):

$$\begin{aligned} B_w &= (\text{cycles/frame})(F_R) \\ &= (\text{cycles/line})(N_L)(F_R) \\ &= \frac{(0.5)(\text{aspect ratio})(F_R)(N_L)(R_H)}{0.84} \\ &= (0.8)(F_R)(M_L)(R_H), \end{aligned} \tag{3}$$

where

B_W = system bandwidth,
F_R = number of frames transmitted per second (fps),
N_L = number of scanning lines per frame,
R_H = horizontal resolution (lines), proportional to pixel resolution.

The National Television Systems Committee (NTSC) aspect ratio is 4/3, the constant 0.5 is the ratio of the number of cycles to the number of lines, and the factor 0.84 is the fraction of the horizontal scanning interval that is devoted to signal transmission.

The NTSC transmission standard used for television broadcasts in the United States has the following parameter values: $F_R = 29.97$ fps, $N_L = 525$ lines, and $R_H = 340$ lines. This yields a video system bandwidth B_W of 4.2 MHz for the NTSC broadcast system. In order to transmit a color digital video signal, the digital pixel format must be defined. The digital color pixel is made of three components: one luminance (Y) component occupying 8 bits, and two color difference components (U and V) each requiring 8 bits. The NTSC picture frame has 720 × 480 × 2 total luminance and color pixels. In order to transmit this information for an NTSC broadcast system at 29.97 frames/s, the following bandwidth is required:

$$\begin{aligned} \text{Digital } B_W \simeq \tfrac{1}{2}\text{bitrate} &= \tfrac{1}{2}(29.97 \text{ fps}) \times (24 \text{ bits/pixel}) \\ &\quad \times (720 \times 480 \times 2 \text{ pixels/frame}) \\ &= 249 \text{ MHz}. \end{aligned}$$

This represents an increase of ~59 times the available system bandwidth, and ~41 times the full transmission channel bandwidth (6 MHz) for current NTSC signals. HDTV picture resolution requires up to three times more raw bandwidth than this example! (Two transmission channels totaling 12 MHz are allocated for terrestrial HDTV transmissions.) It is clear from this example that terrestrial television broadcast systems will have to use digital transmission and digital video compression to achieve the overall bitrate reduction and image quality required for HDTV signals.

The example not only points out the significant bandwidth requirements for digital video information, but also indirectly brings up the issue of digital video quality requirements. The tradeoff between bitrate and quality or distortion is a fundamental issue facing the design of video compression systems. To this end, it is important to fully characterize an application's video communications requirements before designing or selecting an appropriate video compression system. Factors that should be considered in the design and selection of a video compression system include the following items.

1. Video characteristics: video parameters such as the dynamic range, source statistics, pixel resolution, and noise content can affect the performance of the compression system.
2. Transmission requirements: transmission bitrate requirements determine the power of the compression system. Very high transmission bandwidth, storage capacity, or quality requirements may necessitate lossless compression. Conversely, extremely low bitrate requirements may dictate compression systems that trade off image quality for a large compression ratio. *Progressive transmission* is a key issue for selection of the compression system. It is generally used when the transmission bandwidth exceeds the compressed video bandwidth. Progressive coding refers to a multiresolution, hierarchical, or subband encoding of the video information. It allows for transmission and reconstruction of each resolution independently from low to high resolution. In addition, channel errors affect system performance and the quality of the reconstructed video. Channel errors can affect the bit stream randomly or in burst fashion. The channel error characteristics can have different effects on different encoders, and they can range from local to global anomalies. In general, transmission error correction codes (ECC) are used to mitigate the effect of channel errors, but awareness and knowledge of this issue is important.
3. Compression system characteristics and performance: the nature of video applications makes many demands on the video compression system. Interactive video applications such as videoconferencing demand that the video compression systems have symmetric capabilities. That is, each participant in the interactive video session must have the same video encoding and decoding capabilities, and the system performance requirements must be met by both the encoder and decoder. In contrast, television broadcast video has significantly greater performance requirements at the transmitter because it has the responsibility of providing real-time high quality compressed video that meets the transmission channel capacity. Digital video system implementation requirements can vary significantly. Desktop televideo conferencing can be implemented by using software encoding and decoding, or it may require specialized hardware and transmission capabilities to provide a high-quality performance. The characteristics of the application will dictate the suitability of the video compression algorithm for particular system implementations.

The importance of the encoder and system implementation decision cannot be overstated; system architectures and performance capabilities are changing at a rapid pace and the choice of the best solution requires careful analysis of the all possible system and encoder alternatives.

4. Rate-distortion requirements: the rate-distortion requirement is a basic consideration in the selection of the video encoder. The video encoder must be able to provide the bitrate(s) and video fidelity (or range of video fidelity) required by the application. Otherwise, any aspect of the system may not meet specifications. For example, if the bitrate specification is exceeded in order to support a lower MSE, a larger than expected transmission error rate may cause a catastrophic system failure.
5. Standards requirements: video encoder compatibility with existing and future standards is an important consideration if the digital video system is required to interoperate with existing or future systems. A good example is that of a desktop videoconferencing application supporting a number of legacy video compression standards. This results in requiring support of the older video encoding standards on new equipment designed for a newer incompatible standard. Videoconferencing equipment not supporting the old standards would not be capable or as capable to work in environments supporting older standards.

These factors are displayed in Table 1 to demonstrate video compression system requirements for some common video communications applications. The video compression system designer at a minimum should consider these factors in making a determination about the choice of video encoding algorithms and technology to implement.

4 Digital Video Signals and Formats

Video compression techniques make use of signal models order to be able to utilize the body of digital signal ana sis/processing theory and techniques that have been develop over the past fifty or so years. The design of a video compre sion system, as represented by the generalized model introdu in Section 2, requires a knowledge of the signal characteristi and the digital processes that are used to create the digital vid signal. It is also highly desirable to understand video disp systems, and the behavior of the HVS.

4.1 Sampling of Analog Video Signals

Digital video information is generated by sampling the *inten sity* of the original continuous analog video signal $I(x, y,$ in three dimensions. The spatial component of the video sig nal is sampled in the horizontal and vertical dimensions $(x, y$ and the temporal component is sampled in the time dimensio (t). This generates a series of digital images or image sequen $I(i, j, k)$. Video signals that contain colorized information a usually decomposed into three parameters (YC_rC_b, YUV, RGB whose intensities are likewise sampled in three dimensions. Th sampling process inherently quantizes the video signal due t the digital word precision used to represent the intensity value Therefore the original analog signal can never be reprodu exactly, but for all intents and purposes, a high-quality digit video representation can be reproduced with arbitrary closene to the original analog video signal. The topic of video samplin and interpolation is discussed in Chapter 7.2.

An important result of sampling theory is the *Nyquist sam pling theorem*. This theorem defines the conditions under whic

TABLE 1 Digital video application requirements

Application	Bitrate Req.	Distortion Req.	Transmission Req.	Computational Req.	Standards Req.
Network video on demand	1.5 Mbps 10 Mbps	High medium	Internet 100 Mbps LAN	MPEG-1 MPEG-2	MPEG-1 MPEG-2 MPEG-7
Video phone	64 Kbps	High distortion	ISDN p × 64	H.261 encoder H.261 decoder	H.261
Desktop multimedia video CDROM	1.5 Mbps	High distortion to medium	PC channel	MPEG-1 decoder	MPEG-1 MPEG-2 MPEG-7
Desktop LAN videoconference	10 Mbps	Medium distortion	Fast ethernet 100 Mbps	Hardware decoders	MPEG-2, H.261
Desktop WAN videoconference	1.5 Mbps	High distortion	Ethernet	Hardware decoders	MPEG-1, MPEG-4, H.263
Desktop dial-up videoconference	64 Kbps	Very high distortion	POTS and internet	Software decoder	MPEG-4, H.263
Digital satellite television	10 Mbps	Low distortion	Fixed service satellites	MPEG-2 decoder	MPEG-2
HDTV	20 Mbps	Low distortion	12-MHz terrestrial link	MPEG-2 decoder	MPEG-2
DVD	20 Mbps	Low distortion	PC channel	MPEG-2 decoder	MPEG-2

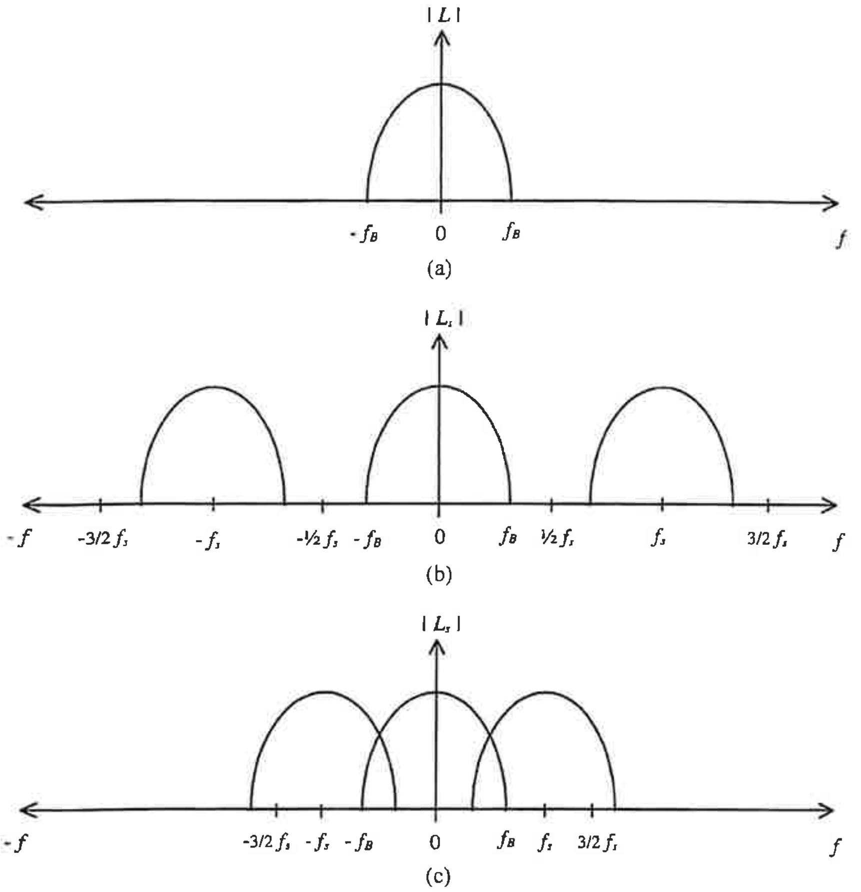


FIGURE 2 Nyquist sampling theorem, with magnitudes of Fourier spectra for (a) input l; (b) sampled input l_s, with $f_s > 2f_B$; (c) sampled input l_s, with $f_s < 2f_B$.

sampled analog signals can be "perfectly" reconstructed. If these conditions are not met, the resulting digital signal will contain *aliased* components which introduce artifacts into the reconstruction. The Nyquist conditions are depicted graphically for the one dimensional case in Fig. 2.

The one dimensional signal l is sampled at rate f_s. It is bandlimited (as are all real-world signals) in the frequency domain with an upper frequency bound of f_B. According to the Nyquist sampling theorem, if a bandlimited signal is sampled, the resulting *Fourier* spectrum is made up of the original signal spectrum $|L|$ plus replicates of the original spectrum spaced at integer multiples of the sampling frequency f_s. Diagram (a) in Fig. 2 depicts the magnitude $|L|$ of the Fourier spectrum for l. The magnitude of the *Fourier* spectrum $|L_s|$ for the sampled signal l_s is shown for two cases. Diagram (b) presents the case where the original signal l can be reconstructed by recovering the central spectral island. Diagram (c) displays the case where the Nyquist sampling criteria has not been met and spectral overlap occurs. The spectral overlap is termed *aliasing* and occurs when $f_s < 2f_B$. When $f_s > 2f_B$, the original signal can be reconstructed by using a low-pass digital filter whose passband is designed to recover $|L|$. These relationships provide a basic framework for the analysis and design of digital signal processing systems.

Two-dimensional or spatial sampling is a simple extension of the one-dimensional case. The Nyquist criteria has to be obeyed in both dimensions; i.e., the sampling rate in the horizontal direction must be two times greater than the upper frequency bound in the horizontal direction, and the sampling rate in the vertical direction must be two times greater than the upper frequency bound in the vertical direction. In practice, spatial sampling grids are square so that an equal number of samples per unit length in each direction are collected. *Charge coupled devices* (CCDs) are typically used to spatially sample analog imagery and video. The sampling grid spacing of these devices is more than sufficient to meet the Nyquist criteria for most resolution and application requirements. The electrical characteristics of CCDs have a greater effect on the image or video quality than its sampling grid size.

Temporal sampling of video signals is accomplished by capturing a spatial or image frame in the time dimension. The temporal samples are captured at a uniform rate of ~60 fields/s for NTSC television and 24 fps for a motion film recording. These sampling rates are significantly smaller than the spatial sampling rate. The maximum temporal frequency that can be reconstructed according to the Nyquist frequency criteria is 30 Hz in the case of television broadcast. Therefore any rapid intensity change (caused, for instance, by a moving edge) between two successive frames will cause aliasing because the harmonic frequency content of such a steplike function exceeds the Nyquist frequency. Temporal aliasing of this kind can be greatly mitigated in CCDs by the use of low-pass temporal filtering to remove the high-frequency content. *Photoconductor storage tubes* are used for recording broadcast television signals. They are analog scanning devices whose electrical characteristics filter the high-frequency temporal content and minimize temporal aliasing. Indeed, motion picture film also introduces low-pass filtering when capturing image frames. The exposure speed and the response speed of the photo chemical film combine to mitigate high-frequency content and temporal aliasing. These factors cannot completely stop temporal aliasing, so intelligent use of video recording devices is still warranted. That is, the main reason movie camera panning is done very slowly is to minimize temporal aliasing.

In many cases in which fast motions or moving edges are not well resolved because of temporal aliasing, the HVS will interpolate such motion and provide its own perceived reconstruction. The HVS is very tolerant of temporal aliasing because it uses its own knowledge of natural motion to provide motion estimation and compensation to the image sequences generated by temporal sampling. The combination of temporal filtering in sampling systems and the mechanisms of human visual perception reduces the effects of temporal aliasing such that temporal undersampling (sub-Nyquist sampling) is acceptable in the generation of typical image sequences intended for general purpose use.

4.2 Digital Video Formats

Sampling is the process used to create the image sequences used for video and digital video applications. Spatial sampling and quantization of a natural video signal digitizes the image plane into a two-dimensional set of digital pixels that define a digital image. Temporal sampling of a natural video signal creates a sequence of image frames typically used for motion pictures and television. The combination of spatial and temporal sampling creates a sequence of digital images termed digital video. As described earlier, the digital video signal intensity is defined as $I(i, j, k)$, where $0 \leq i \leq M$, $0 \leq j \leq N$ are the horizontal and vertical spatial coordinates, and $0 \leq k$ is the temporal coordinate.

The standard digital video formats introduced here are used in the broadcast for both analog and digital television, as well as computer video applications. Composite television signal digital broadcasting formats are introduced here because of their use in video compression standards, digital broadcasting, and standards format conversion applications. Knowledge of these digital video formats provides background for understanding the international video compression standards developed by the ITU and the ISO. These standards contain specific recommendations for use of the digital video formats described here.

TABLE 2 Digital composite television parameters

Description	NTSC	PAL
Analog video bandwidth (MHz)	4.2	5.0
Aspect ratio; hor. size/vert. size	4/3	4/3
Frames/s	29.97	25
Lines/s	525	625
Interlace ratio; fields:frames	2:1	2:1
Subcarrier frequency (MHz)	3.58	4.43
Sampling frequency (MHz)	14.4	17.7
Samples/active line	757	939
Bitrate (Mbps)	114.5	141.9

Composite television digital video formats are used for the digital broadcasting, SMPTE digital recording, and conversion of television broadcasting formats. Table 2 contains both analog and digital system parameters for the NTSC and *Phase Alternating Lines* (PAL) composite broadcast formats.

Component television signal digital video formats have been defined by the *International Consultative Committee for Radio* (CCIR) Recommendation 601. It is based on component video with one luminance (Y) and two color difference signals (C_r and C_b). The raw bitrate for the CCIR 601 format is 162 Mbps. Table 3 contains important systems parameters of the CCIR 601 digital video studio component recommendation for both NTSC and PAL/SECAM (*Sequentiel Couleur avec Memoire*).

The ITU Specialist Group (SGXV) has recommended three formats that are used in the ITU H.261, H.263, and ISO MPEG video compression standards. They are the *Standard Input Format* (SIF), *Common Interchange Format* (CIF), and the low bit-rate version of CIF, called *Quarter CIF* (QCIF). Together, these formats describe a comprehensive set of digital video formats that are widely used in current digital video applications. CIF and QCIF support the NTSC and PAL video formats using the

TABLE 3 Digital video component television parameters for CCIR 601

Description	NTSC	PAL/SECAM
Luminance channel		
Analog video bandwidth (MHz)	5.5	5.5
Sampling frequency (MHz)	13.5	13.5
Samples/active line	710	716
Bitrate (Mbps)	108	108
Color difference channels		
Analog video bandwidth (MHz)	2.2	2.2
Sampling frequency (MHz)	6.75	6.75
Samples/active line	355	358
Bitrate (Mbps)	54	54

TABLE 4 SIF, CIF, and QCIF digital video formats

Description	SIF NTSC/PAL	CIF	QCIF
Horizontal resolution (Y) pixels	352	360(352)	180(176)
Vertical resolution (Y) pixels	240/288	288	144
Horizontal resolution (C_r, C_b) pixels	176	180(176)	90(88)
Vertical resolution (C_r, C_b) pixels	120/144	144	72
Bits/pixel (bpp)	8	8	8
Interlace fields:frames	1:1	1:1	1:1
Frame rate (fps)	30	30, 15, 10, 7.5	30, 15, 10, 7.5
Aspect ratio; hor. size/ vert. size	4:3	4:3	4:3
Bitrate (Y) Mbps @ 30 fps	20.3	24.9	6.2
Bitrate (U, V) Mbps @ 30 fps	10.1	12.4	3.1

same parameters. The SIF format defines different vertical resolution values for NTSC and PAL. The CIF and QCIF formats also support the H.261 modified parameters. The modified parameters are integer multiples of eight in order to support the 8 × 8 pixel two-dimensional DCT operation. Table 4 lists this set of digital video standard formats. The modified H.261 parameters are listed in parentheses.

5 Video Compression Techniques

Video compression systems generally comprise two modes that reduce information redundancy in the spatial and the temporal domains. Spatial compression and quantization operates on a single image block, making use of the local image characteristics to reduce the bitrate. The spatial encoder also includes a VLC inserted after the quantization stage. The VLC stage generates a lossless encoding of the quantized image block. Lossless coding is discussed in Chapter 5.1. Temporal domain compression makes use of optical flow models (generally in the form of block-matching motion estimation methods) to identify and mitigate temporal redundancy.

This section presents an overview of some widely accepted encoding techniques used in video compression systems. *Entropy Encoders* are lossless encoders that are used in the VLC stage of a video compression system. They are best used for information sources that are *memoryless* (sources in which each value is independently generated), and they try to minimize the bitrate by assigning variable length codes for the input values according to the input *probability density function* (pdf). *Predictive coders* are suited to information sources that have memory, i.e., a source in which each value has a statistical dependency on some number of previous and/or adjacent values. Predictive coders can produce a new source pdf with significantly less statistical variation and entropy than the original. The transformed source can then be fed to a VLC to reduce the bitrate. Entropy and predictive coding are good examples for presenting the basic concepts of statistical coding theory.

Block transformations are the major technique for representing spatial information in a format that is highly conducive to quantization and VLC encoding. Block transforms can provide a coding gain by packing most of the block energy into a fewer number of coefficients. The *quantization* stage of the video encoder is the central factor in determining the rate-distortion characteristics of a video compression system. It quantizes the block transform coefficients according to the bitrate and distortion specifications. Motion compensation takes advantage of the significant information redundancy in the temporal domain by creating current frame predictions based upon block matching motion estimates between the current and previous image frames. Motion compensation generally achieves a significant increase in the video coding efficiency over pure spatial encoding.

5.1 Entropy and Predictive Coding

Entropy coding is an excellent starting point in the discussion of coding techniques because it makes use of many of the basic concepts introduced in the discipline of *Information Theory* or *Statistical Communications Theory* [7]. The discussion of VLC and predictive coders requires the use of *information source* models to lay the statistical foundation for the development of this class of encoder. An information source can be viewed as a process that generates a sequence of symbols from a finite alphabet. Video sources are generated from a sequence of image blocks that are generated from a "pixel" alphabet. The number of possible pixels that can be generated is 2^n, when n is the number of bits per pixel. The order in which the image symbols are generated depends on how the image block is arranged or scanned into a sequence of symbols. Spatial encoders transform the statistical nature of the original image so that the resulting coefficient matrix can be scanned in a manner such that the resulting source or sequence of symbols contains significantly less information content.

Two useful information sources are used in modeling video encoders: the *discrete memoryless source* (DMS), and *Markov* sources. VLC coding is based on the DMS model, and the predictive coders are based on the Markov source models. The DMS is simply a source in which each symbol is generated independently. The symbols are *statistically independent* and the source is completely defined by its symbols/events and the set of probabilities for the occurrence for each symbol; i.e., $E = \{e_1, e_2, \ldots, e_n\}$ and the set $\{p(e_1), p(e_2), \ldots, p(e_n)\}$, where n is the number of symbols in the alphabet. It is useful to introduce the concept of entropy at this point. Entropy is defined as the average information content of the information source. The information content of a single event or symbol is defined as

$$I(e_i) = \log \frac{1}{p(e_i)}. \tag{4}$$

The base of the logarithm is determined by the number of states used to represent the information source. Digital information sources use base 2 in order to define the information content using the number of bits per symbol or bitrate. The entropy of a digital source is further defined as the average information content of the source, i.e.,

$$H(E) = \sum_{i=1}^{n} p(e_i) \log_2 \frac{1}{p(e_i)}$$
$$= -\sum_{i=1}^{n} p(e_i) \log_2 p(e_i) \text{ bits/symbol.} \qquad (5)$$

This relationship suggests that the average number of bits per symbol required to represent the information content of the source is the entropy. The *noiseless source coding theorem* states that a source can be encoded with an average number of bits per source symbol that is arbitrarily close to the source entropy. So called entropy encoders seek to find codes that perform close to the entropy of the source. *Huffman* and *arithmetic* encoders are examples of entropy encoders.

Modified Huffman coding [8] is commonly used in the image and video compression standards. It produces well performing variable length codes without significant computational complexity. The traditional Huffman algorithm is a two-step process that first creates a table of source symbol probabilities and then constructs codewords whose lengths grow according to the decreasing probability of a symbol's occurrence. Modified versions of the traditional algorithm are used in the current generation of image and video encoders. The H.261 encoder uses two sets of static Huffman codewords (one each for AC and DC DCT coefficients). A set of 32 codewords is used for encoding the AC coefficients. The zigzag scanned coefficients are classified according to the zero coefficient run length and first nonzero coefficient value. A simple table lookup is all that is then required to assign the codeword for each classified pair.

Markov and *random field* source models (discussed in Chapter 4.2) are well suited to describing the source characteristics of natural images. A Markov source has memory of some number of preceding or adjacent events. In a natural image block, the value of the current pixel is dependent on the values of some the surrounding pixels because they are part of the same object, texture, contour, etc. This can be modeled as an mth order Markov source, in which the probability of source symbol e_i depends on the last m source symbols. This dependence is expressed as the probability of occurrence of event e_i conditioned on the occurrence of the last m events, i.e., $p(e_i \mid e_{i-1}, e_{i-2}, \ldots, e_{i-m})$. The Markov source is made up of all possible n^m states, where n is the number of symbols in the alphabet. Each state contains a set of up to n conditional probabilities for the possible transitions between the current symbol and the next symbol. The *differential pulse code modulation* (DPCM) predictive coder makes use of the Markov source model. DPCM is used in the MPEG-1 and H.261 standards to encode the set of quantized DC coefficients generated by the discrete cosine transforms.

The DPCM predictive encoder modifies the use of the Markov source model considerably in order to reduce its complexity. It does not rely on the actual Markov source statistics at all, and it simply creates a linear weighting of the last m symbols (mth order) to predict the next state. This significantly reduces the complexity of using Markov source prediction at the expense of an increase in the bitrate. DPCM encodes the differential signal d between the actual value and the predicted value, i.e., $d = e - \hat{e}$, where the prediction $\hat{e}$ is a linear weighting of m previous values. The resulting differential signal d generally has reduced entropy as compared to the original source. DPCM is used in conjunction with a VLC encoder to reduce the bitrate. The simplicity and entropy reduction capability of DPCM makes it a good choice for use in real-time compression systems. Third order predictors ($m = 3$) have been shown to provide good performance on natural images [9].

5.2 Block Transform Coding: The Discrete Cosine Transform

Block transform coding is widely used in image and video compression systems. The transforms used in video encoders are *unitary*, which means that the transform operation has a inverse operation that uniquely reconstructs the original input. The DCT successively operates on 8 × 8 image blocks, and it is used in the H.261, H.263, and MPEG standards. Block transforms make use of the high degree of correlation between adjacent image pixels to provide *energy compaction* or coding gain in the transformed domain. The *block transform coding gain*, G_{TC}, is defined as the ratio of the arithmetic and geometric means of the transformed block variances, i.e.,

$$G_{TC} = 10 \log_{10} \left[\frac{\frac{1}{N} \sum_{i=0}^{N-1} \sigma_i^2}{\left(\prod_{i=0}^{N-1} \sigma_i^2 \right)^{1/N}} \right], \qquad (6)$$

where the transformed image block contains N subbands, and σ_i^2 is the variance of each block subband i, for $0 \leq i \leq N-1$. G_{TC} also measures the gain of block transform coding over PCM coding. The coding gain generated by a block transform is realized by packing most the original signal energy content into a small number of transform coefficients. This results in a lossless representation of the original signal that is more suitable for quantization. That is, there may be many transform coefficients containing little or no energy that can be completely eliminated. Spatial transforms should also be orthonormal, i.e., generate uncorrelated coefficients, so that simple scalar quantization can be used to quantize the coefficients independently.

The *Karhunen–Loève transform* (KLT) creates uncorrelated coefficients, and it is optimal in the energy packing sense. But the KLT is not widely used in practice. It requires the calculation of the image block covariance matrix so that its unitary orthonormal eigenvector matrix can be used to generate the KLT coefficients. This calculation (for which no fast algorithms exist), and the transmission of the eigenvector matrix, is required for every transformed image block.

The DCT is the most widely used block transform for digital image and video encoding. It is an orthonormal transform, and it has been found to perform close to the KLT [10] for first-order Markov sources. The DCT is defined on an 8 × 8 array of pixels,

$$F(u, v) = \frac{1}{4} C_u C_v \sum_{i=0}^{7} \sum_{j=0}^{7} f(i, j) \cos\left(\frac{(2i+1)u\pi}{16}\right) \times \cos\left(\frac{(2j+1)v\pi}{16}\right), \tag{7}$$

and the inverse DCT (IDCT) is defined as

$$f(i, j) = C_u C_v \sum_{u=v}^{7} \sum_{v=0}^{7} F(u, v) \cos\left(\frac{(2i+1)u\pi}{16}\right) \times \cos\left(\frac{(2j+1)v\pi}{16}\right), \tag{8}$$

with

$$C_u = \frac{1}{\sqrt{2}} \quad \text{for } u = 0, \quad C_u = 1 \text{ otherwise,}$$
$$C_v = \frac{1}{2} \quad \text{for } v = 0, \quad C_v = 1 \text{ otherwise}$$

where i and j are the horizontal and vertical indices of the 8 × 8 spatial array, and u and v are the horizontal and vertical indices of the 8 × 8 coefficient array. The DCT is the chosen method for image transforms for a couple of important reasons. The DCT has fast $O(n \log n)$ implementations using real calculations. It is even simpler to compute than the DFT because it does not require the use of complex numbers.

The second reason for its success is that the reconstructed input of the IDCT tends not to produce any significant discontinuities at the block edges. Finite discrete transforms create a reconstructed signal that is periodic. Periodicity in the reconstructed signal can produce discontinuities at the periodic edges of the signal or pixel block. The DCT is not as susceptible to this behavior as the DFT. Since the cosine function is real and even, i.e., $\cos(x) = \cos(-x)$, and the input $F(u, v)$ is real, the IDCT generates a function that is even and periodic in $2n$, where n is the length of the original sequence. In contrast, the IDFT produces a reconstruction that is periodic in n, but and necessarily even. This phenomenon is illustrated in Fig .3 for the one-dimensional signal $f(i)$.

The original finite sequence $f(i)$ depicted in part Fig. 3(a) is transformed and reconstructed in Fig. 3(b) by using the DFT–IDFT transform pairs, and in Fig. 3(c) by using the DCT–IDCT transform pairs. The periodicity of the IDFT in Fig. 3(b) is five samples long and illustrates the discontinuity introduced by the discrete transform. The periodicity of the IDCT in Fig. 3(c) is 10 samples long, as a result of the evenness of the DCT operation. Discontinuities introduced by the DCT are generally less severe than those of the DFT. The importance of this property of the DCT is that reconstruction errors and blocking artifacts are less severe in comparison to those of the DFT. Blocking artifacts are visually striking and occur because of the loss of high-frequency components that are either quantized or eliminated from the DCT coefficient array. The DCT minimizes blocking artifacts as compared to the DFT because it does not introduce the same level of reconstruction discontinuities at the block edges. Figure 4 depicts blocking artifacts introduced by gross quantization of the DCT coefficients.

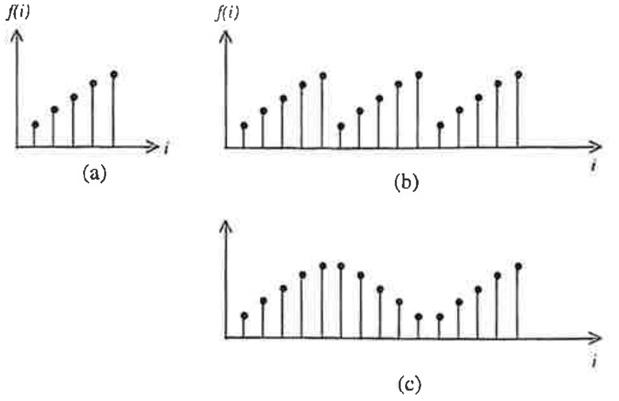


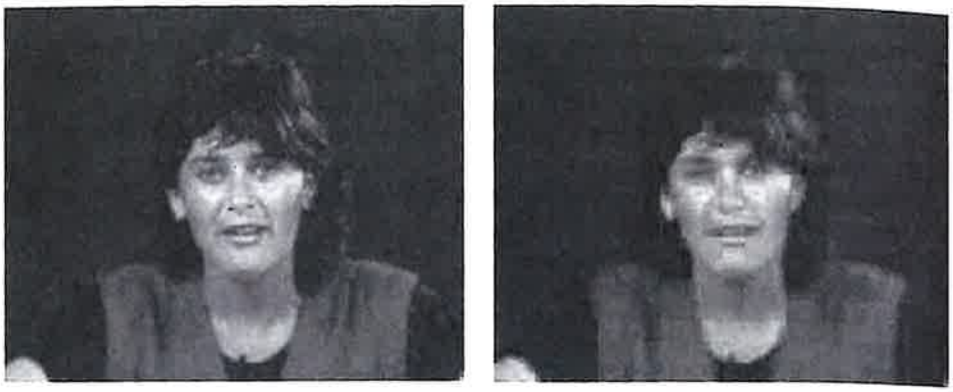
FIGURE 3 Reconstruction periodicity of DFT vs. DCT: (a) original sequence; (b) DFT reconstruction; (c) DCT reconstruction.

This section ends with an example of the energy packing capability of the DCT. Figure 5 depicts the DCT transform operation. The original 8 × 8 image subblock from the Lena image is displayed in Fig. 5(a), and the DCT transformed coefficient array is displayed in Fig. 5(b).

The original image subblock in Fig. 5(a) contains large values in every position, and is not very suitable for spatial compression in this format. The coefficient matrix (b) concentrates most of the signal energy in the top left quadrant. The signal frequency coordinates $(u, v) = (0, 0)$ start at the upper left position. The DC component equals 1255 and contains the vast majority of the signal energy by itself. This dynamic range and concentration of energy should yield a significant reduction in nonzero values and bitrate after the coefficients are quantized.

5.3 Quantization

The quantization stage of the video encoder creates a lossy representation of the input. The input, as discussed earlier, should be



(a) (b)

FIGURE 4 Severe blocking artifacts introduced by gross quantization of DCT coefficients: (a) original, (b) reconstructed. (See color section, p. C–27.)

conditioned with a particular method of quantization in mind. Vice versa, the quantizer should be well matched to the characteristics of the input in order to meet or exceed the rate-distortion performance requirements. As is always the case, the quantizer has an effect on system performance that must be taken under consideration. Simple scalar versus vector quantization implementations can have significant system performance implications.

Scalar and vector are the two major types of quantizers. These can be further classified as memoryless or containing memory, and symmetric or nonsymmetric. Scalar quantizers control the values taken by a single variable. The quantizer defined by the MPEG-1 encoder scales the DCT transform coefficients. Vector quantizers operate on multiple variables, i.e., a vector of variables, and become very complex as the number of variables increases. This discussion will introduce the reader to the basic scalar and vector quantizer concepts that are relevant to image and video encoding.

$$f(i,j)=\begin{bmatrix} 136 & 141 & 143 & 153 & 152 & 154 & 154 & 156 \\ 143 & 150 & 153 & 156 & 160 & 156 & 155 & 155 \\ 149 & 155 & 161 & 163 & 158 & 155 & 156 & 155 \\ 158 & 161 & 162 & 161 & 160 & 158 & 160 & 157 \\ 157 & 161 & 160 & 162 & 161 & 157 & 154 & 155 \\ 160 & 160 & 161 & 160 & 160 & 156 & 156 & 156 \\ 160 & 161 & 160 & 161 & 161 & 157 & 157 & 156 \\ 162 & 162 & 161 & 161 & 162 & 157 & 157 & 157 \end{bmatrix}$$

(a)

$$F(u,v)=\begin{bmatrix} 1255 & -8 & -9 & -6 & 1 & -1 & -3 & 1 \\ -26 & -20 & -5 & 4 & -1 & 1 & 0 & 1 \\ -9 & -5 & 1 & -1 & 0 & 0 & -1 & 0 \\ -6 & -2 & 0 & 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 2 & 0 & -1 & -1 & 0 \\ -2 & 1 & 2 & 0 & 1 & 1 & 0 & -1 \\ -1 & 0 & 0 & -2 & 0 & 0 & 1 & -1 \\ 1 & 0 & -1 & -2 & 0 & 1 & -1 & 0 \end{bmatrix}$$

(b)

FIGURE 5 8 × 8 DCT: (a) original lena 8 × 8 image subblock; (b) DCT coefficients.

The uniform scalar quantizer is the most fundamental scalar quantizer. It possesses a nonlinear staircase input–output characteristic that divides the input range into output levels of equal size. In order for the quantizer to effectively reduce the bitrate, the number of output values should be much smaller than the number of input values. The reconstruction values are chosen to be at the midpoint of the output levels. This choice is expected to minimize the reconstruction MSE when the quantization errors are uniformly distributed. The quantizers specified in the H.261, H.263, MPEG-1, and MPEG-2 video coders are *nearly* uniform. They have constant step sizes except for the larger *dead-zone* area (the input range for which the output is zero).

Non-uniform quantization is typically used for non-uniform input distributions, such as natural image sources. The scalar quantizer that produces the minimum MSE for a non-uniform input distribution will have non-uniform steps. Compared with the uniform quantizer, the non-uniform quantizer has increasingly better MSE performance as the number of quantization steps increases. The Lloyd–Max [11] is a scalar quantizer design that utilizes the input distribution to minimize the MSE for a given number of output levels. The Lloyd–Max places the reconstruction levels at the centroids of the adjacent input quantization steps. This minimizes the total absolute error within each quantization step based upon the input distribution..

Vector quantizers (discussed in Chapter 5.3) decompose the input into a length n vector. An image for instance, can be divided into $M \times N$ blocks of n pixels each, or the image block can be transformed into a block of transform coefficients. The resulting vector is created by scanning the two-dimensional block elements into a vector of length n. A vector $\mathbf{X}$ is quantized by choosing a codebook vector representation $\hat{\mathbf{X}}$ that is its "closest match." The closest match selection can be made by minimizing an error measure, i.e., choose $\hat{\mathbf{X}} = \hat{\mathbf{X}}_i$ such that the MSE over all codebook vectors is minimized:

$$\hat{\mathbf{X}} = \hat{\mathbf{X}}_i: \quad \min_i \mathrm{MSE}(\mathbf{X}, \hat{\mathbf{X}}_i) = \min_i \frac{1}{n} \sum_{j=1}^{n} (x_j - \hat{x}_j)^2. \tag{9}$$

The index i of the vector $\hat{\mathbf{X}}_i$ denotes the codebook entry that is used by the receiver to decode the vector. Obviously the complexity of the decoder is much simpler than the encoder. The size of the codebook dictates both the coding efficiency and reconstruction quality. The raw bitrate of a vector quantizer is

$$\mathrm{bitrate}_{\mathrm{VQ}} = \frac{\log_2 m}{n} \text{ bits/pixel}, \tag{10}$$

where $\log_2 m$ is the number of bits required to transmit the index i of the codebook vector $\hat{\mathbf{X}}_i$. The codebook construction includes two important issues that are pertinent to the performance of the video coder. The set of vectors that are included in the codebook determine the bitrate and distortion characteristics of the reconstructed image sequence. The codebook size and structure determines the search complexity to find the minimum error solution for Eq. (9). Two important VQ codebook designs are the *Linde–Buzo–Gray* (LBG) [12] and TSVQ [13]. The LBG design is based on the Lloyd–Max scalar quantizer algorithm. It is widely used because the system parameters can be generated by the use of an input "training set" instead of the true source statistics. The TSVQ design reduces VQ codebook search time by using m-ary tree structures and searching techniques.

5.4 Motion Compensation and Estimation

Motion compensation [14] is a technique created in the 1960s that used to increase the efficiency of video encoders. Motion compensated video encoders are implemented in three stages. The first stage estimates objective motion (motion estimation) between the previously reconstructed frame and the current frame. The second stage creates the current frame prediction (motion compensation), using the motion estimates and the previously reconstructed frame. The final stage differentially encodes the prediction and the actual current frame as the prediction error. Therefore, the receiver reconstructs the current image only by using the VLC encoded motion estimates and the spatially and VLC encoded prediction error.

Motion estimation and compensation are common techniques used to encode the temporal aspect of a video signal. As discussed earlier, block-based motion compensation and motion estimation techniques used in video compression systems are capable of the largest reduction in the raw signal bitrate. Typical implementations generally outperform pure spatial encodings by a factor of 3 or more. The interframe redundancy contained in the temporal dimension of a digital image sequence accounts for the impressive signal compression capability that can be achieved by video encoders. Interframe redundancy can be simply modeled as static backgrounds and moving foregrounds to illustrate the potential temporal compression that can be realized. Over a short period of time, image sequences can be described as a static background with moving objects in the foreground. If the background does not change between two frames, their difference is zero, and the two background frames can essentially be encoded as one. Therefore the compression ratio increase is proportional to two times the spatial compression achieved in the first frame. In general, unchanging or static backgrounds can realize additive coding gains, i.e.,

$$\text{Static Background Coding Gain} \propto N \bullet (\text{Spatial Compression Ratio of Background Frame}), \tag{11}$$

where N is the number of static background frames being encoded. Static backgrounds occupy a great deal of the image area and are typical of both natural and animated image sequences. Some variation in the background always occurs as a result of random and systematic fluctuations. This tends to reduce the achievable background coding gain.

Moving foregrounds are modeled as nonrotational rigid objects that move independently of the background. Moving objects can be detected by matching the foreground object between two frames. A perfect match results in zero difference between the two frames. In theory, moving foregrounds can also achieve additive coding gain. In practice, moving objects are subject to occlusion, rotational and nonrigid motion, and illumination variations that reduce the achievable coding gain. Motion compensation systems that make use of motion estimation methods leverage both background and foreground coding gain. They provide pure interframe differential encoding when two backgrounds are static; i.e., the computed motion vector is (0,0). The motion estimate computed in the case of moving foregrounds generates the minimum distortion prediction.

Motion estimation is an interframe prediction process falling in two general categories: pel-recursive algorithms [15] and block-matching algorithms (BMAs) [16]. The pel-recursive methods are very complex and inaccurate and restrict their use in video encoders. Natural digital image sequences generally display ambiguous object motion that adversely affects the convergence properties of pel-recursive algorithms. This has led to the introduction of *block-matching motion estimation*, which is tailored for encoding image sequences. Block-matching motion estimation assumes that the objective motion being predicted is rigid and nonrotational. The block size of the BMA for the MPEG, H.261, and H.263 encoders is defined as 16 × 16 luminance pixels. MPEG-2 also supports 16 × 8 pixel blocks.

BMAs predict the motion of a block of pixels between two frames in an image sequence. The prediction generates a pixel displacement or motion vector whose size is constrained by the search neighborhood. The search neighborhood determines the complexity of the algorithm. The search for the best prediction ends when the best block match is determined within the search neighborhood. The best match can be chosen as the minimum

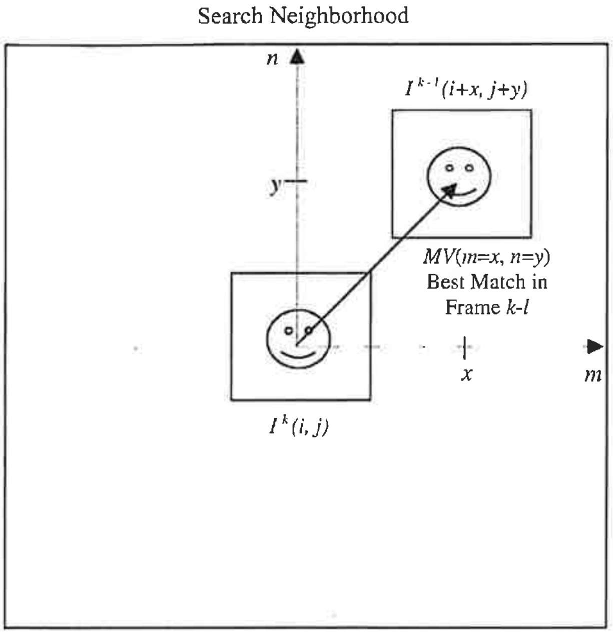


FIGURE 6 Best match motion estimate.

MSE, which for a full search is computed for each block in the search neighborhood:

$$\text{BestMatch}_{\text{MSE}} = \min_{m,n} \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} [I^k(i, j) - I^{k-l}(i + m, j + n)]^2, \quad (12)$$

where k is the frame index, l is the temporal displacement in frames, N is the number of pixels in the horizontal and vertical directions of the image block, i and j are the pixel indices within the image block, and m and n are the indices of the search neighborhood in the horizontal and vertical directions. Therefore the best match motion vector estimate $MV(m = x, n = y)$ is the pixel displacement between the block $I^k(i, j)$ in frame k, and the best matched block $I^{k-l}(i + x, j + y)$ in the displaced frame $k - l$. The best match is depicted in Fig. 6.

In cases in which the block motion is not uniform or if the scene changes, the motion estimate may in fact increase the bitrate over a spatial encoding of the block. In the case in which the motion estimate is not effective, the video encoder does not use the motion estimate and encodes the block by using the spatial encoder.

The search space size determines the complexity of the motion estimation algorithm. Full search methods are costly and are not generally implemented in real-time video encoders. Fast searching techniques can considerably reduce computational complexity while maintaining good accuracy. These algorithms reduce the search process to a few sequential steps in which each subsequent search direction is based upon the results of the current step. The procedures are designed to find local optimal solutions and cannot guarantee selection of the global optimal solution within the search neighborhood. The *logarithmic* search [17] algorithm proceeds in the direction of minimum distortion until the final optimal value is found. Logarithmic searching has been implemented in some MPEG encoders. The *three-step* search [18] is a very simple technique that proceeds along a best match path in three steps in which the search neighborhood is reduced for each successive step. Figure 7 depicts the three-step search algorithm.

A 14 × 14 pixel search neighborhood is depicted. The search area sizes for each step are chosen so that the total search neighborhood can be covered in finding the local minimum. The search areas are square. The length of sides of the search area for step 1 are chosen to be larger than or equal to ½ the length of the range of the search neighborhood (in this example the search area is 8 × 8). The length of the sides are reduced by ½ after each of the first two steps are completed. Nine points for each step are compared by using the matching criteria. These consist of the central point and eight equally spaced points along the perimeter of the search area. The search area for step 1 is centered on the search neighborhood. The search proceeds by centering the search area for the next step over the best match from the previous step. The overall best match is the pixel displacement chosen to minimize the matching criteria in step 3. The total number of required comparisons for the three-step algorithm is 25. That represents an 87% reduction in complexity versus the full search method for a 14 × 14 pixel search neighborhood.

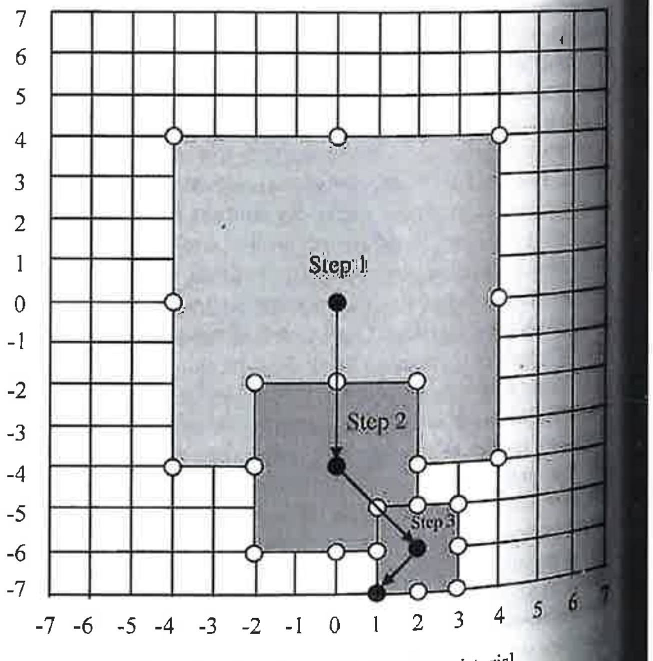


FIGURE 7 Three-step search algorithm pictorial.

6 Video Encoding Standards and H.261

The major internationally recognized video compression standards have been developed by the ISO, the International Electrotechnical Commission (IEC), and the ITU standards organizations.

The MPEG is a working group operating within ISO and IEC. Since starting its activity in 1988, MPEG has produced ISO/IEC 11172 (MPEG-1) and ISO/IEC 13818 (MPEG-2). The MPEG-1 specification was motivated by T1 transmission speeds, the CD-ROM, and the multimedia capabilities of the desktop computer. It is intended for video coding up to the rate of 1.5 Mbps, and it is composed of five sections: system configurations, video coding, audio coding, compliance testing, and software for MPEG-1 coding. The standard does not specify the actual video coding process, but only the syntax and semantics of the bit stream, and the video generation at the receiver. It does not accommodate interlaced video, and it only supports CIF quality format at 25 or 30 fps.

Activity for MPEG-2 was started in 1991. It was targeted for higher bitrates, broadcast video, and a variety of consumer and telecommunications video and audio applications. The syntax and technical contents of the standard were frozen in 1993. It is composed of four parts: systems, video, audio, and conformance testing. MPEG-2 was also recommended by the ITU as H.262.

MPEG is considering more advanced forms of video application interactivity that technology will make possible in the next few years. The MPEG-4 project is targeted to give users the possibility to achieve various forms of interactivity with the audiovisual content of a scene, and to mix synthetic and natural audio and video information in a seamless way. MPEG-4 technology comprises two major parts: a set of coding tools for audiovisual objects, and a syntactic language to describe both the coding tools and the coded objects. From a technical viewpoint, the most notable departure from traditional coding standards will be the possibility for a receiver to download the description of the syntax used to represent the audiovisual information. The visual information will not be restricted to the format of conventional video, i.e., it will not necessarily be frame based. This is expected to produce significant improvements in both encoder efficiency and functionality.

The ITU Recommendation H.261 was adopted in 1990 and specifies a video encoding standard for videoconferencing and videophone services for transmission over the Integrated Services Digital Network (ISDN) at $p \times 64$ Kbps, $p = 1, \ldots, 30$. H.261 describes the video compression methods that were later adopted by the MPEG standards and is presented in the following section. The ITU Experts Group for Very Low Bit-Rate Video Telephony (LBC) has produced the H.263 recommendation for *Public Switched Telephone Networks* (PSTN), which was finalized in December 1995 [18]. It is an extended version of H.261 supporting bidirectional motion compensation and sub-QCIF formats. The encoder is based on hybrid DPCM/DCT coding and improvements targeted to generate bitrates of less than 64 Kbps.

6.1 The H.261 Video Encoder

The H.261 recommendation [3] is targeted at the videophone and videoconferencing application market running on connection-based ISDN at $p \times 64$ kbps, $p = 1, \ldots, 30$. It explicitly defines the encoded bit stream syntax and *decoder*, while leaving the encoder design to be compatible with the decoder specification. The video encoder is required to carry a delay of less than 150 ms so that it can operate in real-time bidirectional videoconferencing applications. H.261 is part of a group of related ITU recommendations that define visual telephony systems. This group includes the following.

1. H.221: defines the frame structure for an audiovisual channel supporting 64–1920 Kbps.
2. H.230: defines frame control signals for audiovisual systems.
3. H.242: defines audiovisual communications protocol for channels supporting up to 2 Mbps.
4. H.261: defines the video encoder/decoder for audiovisual services at $p \times 64$ Kbps.
5. H.320: defines narrow-band audiovisual terminal equipment for $p \times 64$ Kbps transmission.

The H.261 encoder block diagrams are depicted in Fig. 8(a) and 8(b). An H.261 source coder implementation is depicted in Fig. 8(c). The source coder implements the video encoding algorithm that includes the spatial encoder, the quantizer, the temporal prediction encoder, and the VLC. The spatial encoder is defined to use the two-dimensional 8 × 8 pixel block DCT and a nearly uniform scalar quantizer, using up to a possible 31 step sizes to scale the AC and interframe DC coefficients. The resulting quantized coefficient matrix is zigzag scanned into a vector that is variable length coded using a hybrid modified run length and Huffman coder. Motion compensation is optional. Motion estimation is only defined in the forward direction because H.261 is limited to real-time videophone and videoconferencing. The recommendation does not specify the motion estimation algorithm or the conditions for the use of intraframe versus interframe encoding.

The video multiplex coder creates a H.261 bitstream that is based on the data hierarchy described below. The transmission buffer is chosen not to exceed the maximum coding delay of 150 ms, and it is used to regulate the transmission bitrate by means of the coding controller. The transmission coder embeds an ECC into the video bit stream that provides error resilience, error concealment, and video synchronization.

H.261 supports most of the internationally accepted digital video formats. These include CCIR 601, SIF, CIF, and QCIF. These formats are defined for both NTSC and PAL broadcast signals. The CIF and QCIF formats were adopted in 1984 by H.261

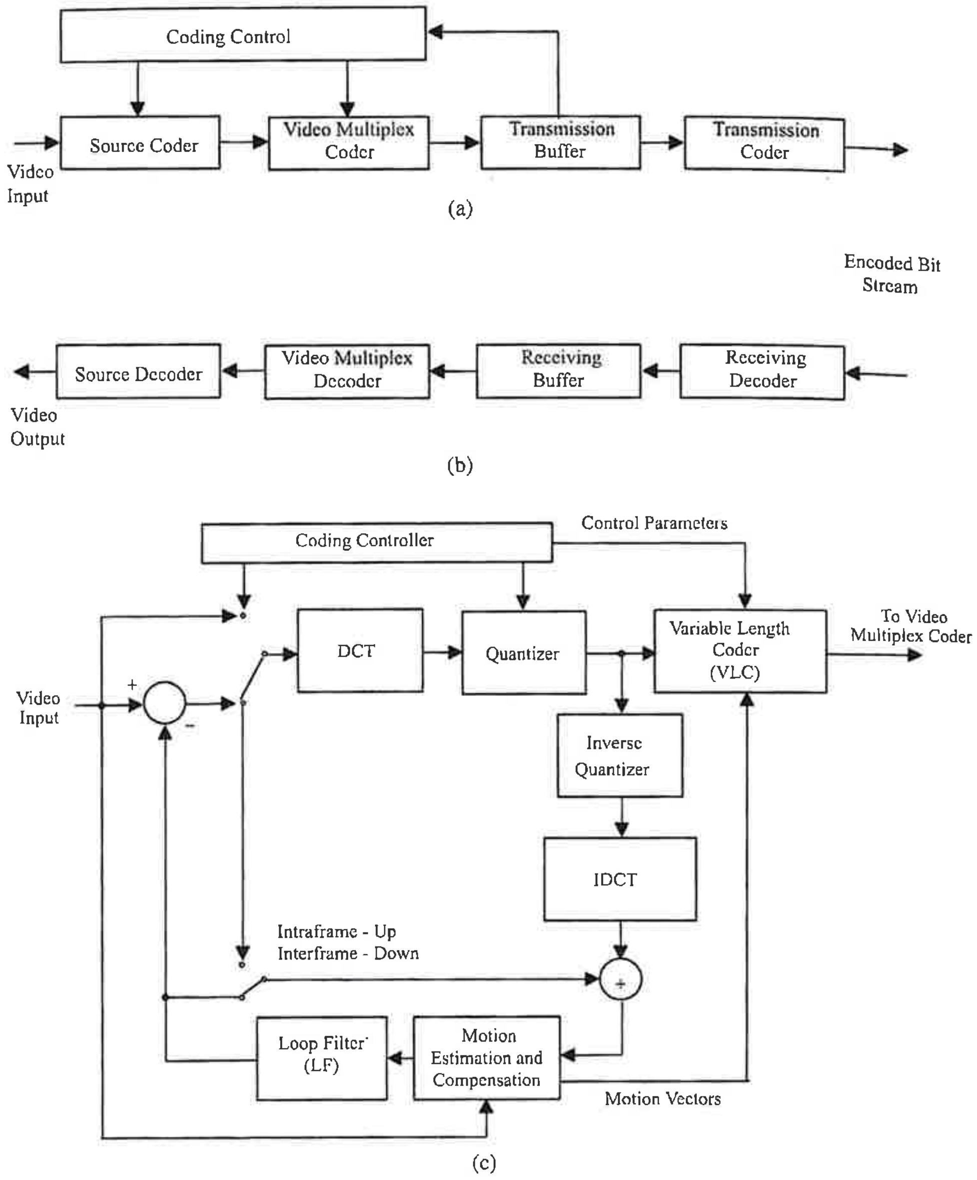


FIGURE 8 ITU-T H.261 block diagrams: (a) video encoder; (b) video decoder; (c) source encoder implementation.

in order to support 525-line NTSC and 625-line PAL/SECAM video formats. The CIF and QCIF operating parameters can be found in Table 4. The raw data rate for 30 fps CIF is 37.3 Mbps and 9.35 Mbps for QCIF. CIF is defined for use in channels in which $p \geq 6$ so that the required compression ratio for 30 fps is smaller than 98:1. CIF and QCIF formats support frame rates of 30, 15, 10, and 7.5 fps, which allows the H.261 encoder to achieve greater coding efficiency by skipping the encoding and transmission of whole frames. H.261 allows zero, one, two, three or more frames to be skipped between transmitted frames.

H.261 specifies a set of encoder protocols and decoder operations that every compatible system must follow. The H.261 *video multiplex* defines the data structure hierarchy that the decoder can interpret unambiguously. The video data hierarchy defined in H.261 is depicted in Fig. 9. They are the picture layer, group of block (GOB) layer, macroblock (MB) layer and the basic (8 × 8) block layer. Each layer is built from the previous or lower layer and contains its associated data payload, and header that describes the parameters used to generate the bit stream. The basic 8 × 8 block is used in intraframe DCT encoding. The MB is the smallest unit for selecting intraframe or interframe encoding modes. It is made up of four adjacent 8 × 8 luminance blocks and two subsampled 8 × 8 color difference blocks (C_B and C_R as defined in Table 4) corresponding to the luminance

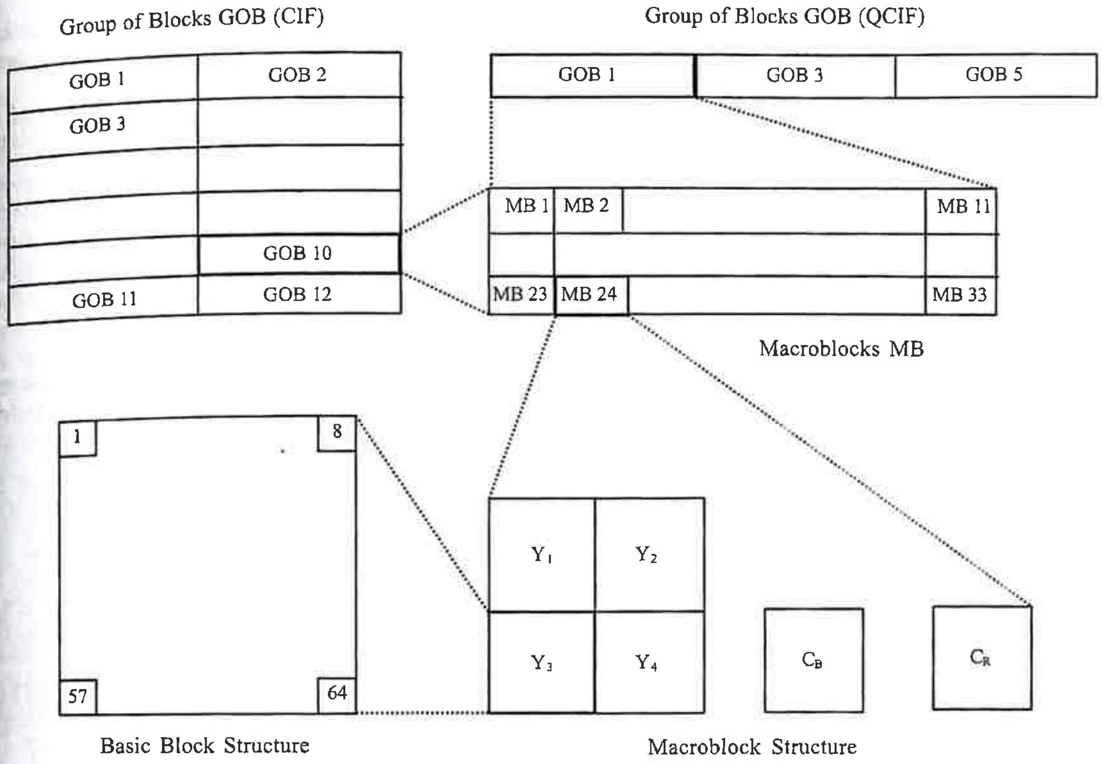


FIGURE 9 H.261 block hierarchy.

blocks. The GOB is made up of 176 × 48 pixels (33 MBs) and is used to construct the 352 × 288 pixel CIF or 176 × 144 pixel QCIF picture layer.

The headers for the GOB and picture layers contain start codes so that the decoder can resynchronize when errors occur. They also contain other relevant information required to reconstruct the image sequence. The following parameters used in the headers of the data hierarchy complete the H.261 video multiplex.

Picture layer:

- Picture start code (PSC), 20-bit synchronization pattern (0000 0000 0000 0001 0000).
- Temporal reference (TR), 5-bit input frame number.
- Type information (PTYPE), indicates source format, CIF = 1 QCIF = 0, and other controls.
- User-inserted bits.

GOB layer:

- Group of blocks start code (GBSC), 16-bit synchronization code (0000 0000 0000 0001).
- Group number (GN), 4-bit address representing the 12 GOBs per CIF frame.
- Quantizer information (GQUANT), indicates one of 31 quantizer step sizes to be used in a GOB unless overridden by Macroblock MQUANT parameter.
- User-inserted bits.

Macroblock layer:

- Macroblock address (MBA) is the position of a MB within a GOB.
- Type information (MTYPE) for one of 10 encoding modes used for the MB. This includes permutations of intraframe, interframe, motion compensation (MC), and loop filtering (LF). A prespecified VLC is used to encode these modes.
- Quantizer (MQUANT), 5-bit normalized quantizer step size from 1–31.
- Motion vector data (MVD), up to 11-bit VLC describing the differential displacement.
- Coded block pattern (CBP), up to 9-bit VLC indicating the location of the encoded blocks in the MB.

Block layer:

- Transform coefficients (TCOEFF) are zigzag scanned and can be 8-bit fixed or up to 13-bit VLC.
- End of block (EOB), symbol.

The H.261 bit stream also specifies transmission synchronization and error code correction by using a BCH code [19] that is capable of correcting 2-bit errors in every 511-bit block. It inserts 18 parity bits for every 493 data bits. A synchronization bit is added to every codeword to be able to detect the BCH codeword boundaries. The transmission synchronization and encoding also operates on the audio and control information specified by the ITU H.320 Recommendation.

The H.261 video compression algorithm depicted in Fig. 8(c) is specified to operate in intraframe and interframe encoding modes. The intraframe mode provides spatial encoding of the 8×8 block and uses the two-dimensional DCT. Interframe mode encodes the prediction error, with motion compensation being optional. The prediction error is optionally DCT encoded. Both modes provide options that effect the performance and video quality of the system. The motion estimate method, mode selection criteria, and block transmission criteria are not specified, although the ITU has published reference models [20, 21] that make particular implementation recommendations. The coding algorithm used in the ITU-T *Reference Model* 8 (RM8) [21] is summarized in three steps, and is followed by an explanation of its important encoding elements.

1. The motion estimator creates a displacement vector for each MB. The motion estimator generally operates on the 16×16 pixel luminance MB. The displacement vector is an integer value between ±15, which is the maximum size of the search neighborhood. The motion estimate is scaled by a factor of 2 and applied to the C_R and C_B component macroblocks.
2. The compression mode for each macroblock is selected by using a minimum error criteria that is based upon the *displaced macroblock difference* (DMD),

$$\mathrm{DMD}\,(i, j, k) = b(i, j, k) - b(i - d_1, j - d_2, k - 1), \tag{13}$$

where b is a 16×16 MB, i and j are its spatial pixel indices, k is the frame index, and d_1 and d_2 are the pixel displacements of the MB in the previous frame. The displacements range from $-15 \leq d_1, d_2 \leq +15$. When d_1 and d_2 are set to zero, the DMD becomes the *macroblock difference* (MD). The compression mode determines the operational encoder elements that are used for the current frame. The H.261 compression modes are depicted in Table 5.

TABLE 5 H.261 MB video compression modes

Mode	MQUANT	MVD	CBP	TCOEFF
Intra				√
Intra	√			√
Inter			√	√
Inter	√		√	√
Inter + MC		√		
Inter + MC		√	√	√
Inter + MC	√	√	√	√
Inter + MC + LF		√		
Inter + MC + LF		√	√	√
Inter + MC + LF	√	√	√	√

3. The *video multiplex coder* processes each macroblock to generate the H.261 video bit stream whose elements are discussed above.

There are five basic MTYPE encoding mode decisions that are carried out in step 2. These are as follows.

- Use intraframe or interframe mode?
- Use motion compensation?
- Use a coded block pattern (CBP)?
- Use loop filtering?
- Change quantization step size MQUANT?

To select the macroblock compression mode, the *variances* (VAR) of the input macroblock, the MD, and the DMD (as determined by the best motion estimate) are compared as follows.

1. If VAR(DBD) < VAR(MD) then interframe + motion compensation (Inter + MC) coding is selected. In this case, the motion vector data (MVD) is transmitted. Table 5 indicates that there are three Inter + MC modes that allow for the transmission of the prediction error (DMD) with or without DCT encoding of some or all of the four 8×8 basic blocks.
2. "VAR input" is defined as the variance of the input macroblock. If VAR input < VAR(DMD) and VAR input < VAR(MD), then the intraframe mode (Intra) is selected. Intraframe mode uses DCT encoding of all four 8×8 basic blocks.
3. If VAR(MD) < VAR(DMD), then interframe mode (Inter) is selected. This mode indicates that the motion vector is zero, and that some or all of the 8×8 prediction error (MD) blocks can be DCT encoded.

The transform coefficient CBP parameter is used to indicate whether a basic block is reconstructed using the corresponding basic block from the previous frame, or if it is encoded and transmitted. In other words, no basic block encoding is used when the block content does not change significantly. The CPB parameter encodes 63 combinations of the four luminance blocks and two color difference blocks using a variable length code. The conditions for using CBP are not specified in the H.261 recommendation.

Motion compensated blocks can be chosen to be low-pass filtered before the prediction error is generated by the feedback loop. This mode is denoted as Inter + MC + LF in Table 5. The low-pass filter is intended to reduce the quantization noise in the feedback loop, as well as the high-frequency noise and artifacts introduced by the motion compensator. H.261 defines loop filtering as optional and recommends a separable two-dimensional spatial filter design, which is implemented by cascading two identical one-dimensional finite impulse response (FIR) filters. The coefficients of the 1-D filter are [1, 2, 1] for pixels inside the block, and [0, 1, 0] (no filtering) for pixels on the block boundary.

The MQUANT parameter is controlled by the state of the transmission buffer in order to prevent overflow or underflow onditions. The dynamic range of the DCT macroblock coefficients extends between [−2047, . . . , 2047]. They are quantized o the range [−127, . . . , 127] using one of the 31 quantizer step izes as determined by the GQUANT paramcter. The step size an even integer in the range of [2, . . . , 62]. GQUANT can e *overridden* at the macroblock layer by MQUANT to clip or xpand the range prescribed by GQUANT so that the transmision buffer is better utilized. The ITU-T RM8 *liquid level control odel* specifies the inspection of 64-Kbit transmission buffers fter encoding 11 macroblocks. The step size of the quantizer hould be increased (decreasing the bitrate) if the buffer is full; ice versa, the step size should be decreased (increasing the birate) if the buffer is empty. The actual design of the rate control lgorithm is not specified.

The DCT macroblock coefficients àre subjected to variable thresholding before quantization. The threshold is designed to ncrease the number of zero valued coefficients, which in turn ncreases the number of the zero run lengths and VLC coding efficiency. The ITU-T RM8 provides an example thresholding algorithm for the H.261 encoder. Nearly uniform scalar quantization sing a dead zone is applied after the thresholding process. All he coefficients in the luminance and chrominance macroblocks re subjected to the same quantizer, except for the intraframe DC oefficient. The intraframe DC coefficient is quantized by using uniform scalar quantizer whose step size is 8. The quantizer ecision levels are not specified, but the reconstruction levels are efined in H.261 as follows.

For case QUANT odd,

$$\begin{aligned}\mathrm{REC\_LEVEL} &= \mathrm{QUANT} \times (2 \times \mathrm{COEFF\_VALUE} + 1),\\ &\quad \text{for } \mathrm{COEFF\_LEVEL} > 0,\\ \mathrm{REC\_LEVEL} &= \mathrm{QUANT} \times (2 \times \mathrm{COEFF\_VALUE} - 1),\\ &\quad \text{for } \mathrm{COEFF\_LEVEL} < 0.\end{aligned}$$

For case QUANT even,

$$\begin{aligned}\mathrm{REC\_LEVEL} &= \mathrm{QUANT} \times (2 \times \mathrm{COEFF\_VALUE} + 1) - 1,\\ &\quad \text{for } \mathrm{COEFF\_LEVEL} > 0,\\ \mathrm{REC\_LEVEL} &= \mathrm{QUANT} \times (2 \times \mathrm{COEFF\_VALUE} - 1) + 1,\\ &\quad \text{for } \mathrm{COEFF\_LEVEL} < 0.\end{aligned}$$

If COEFF_VALUE = 0, then REC_LEVEL = 0, where EC_LEVEL is the reconstruction value, QUANT is ½ the macoblock quantization step size ranging 1–31, and COEFF_VALUE the quantized DCT coefficient.

To increase the coding efficiency, lossless variable length codg is applied to the quantized DCT coefficients. The coefficient atrix is scanned in a zigzag manner in order to maximize the umber of zero coefficient run lengths. The VLC encodes events fined as the combination of a run length of zero coefficients receding a nonzero coefficient, and the value of the nonzero coefficient, i.e., EVENT = (RUN, VALUE). The VLC EVENT tables are defined in [3].

7 Closing Remarks

Digital video compression, although only recently becoming a standardized technology, is strongly based upon the information coding technologies developed over the past 40 years. The large variety of bandwidth and video quality requirements for the transmission and storage of digital video information has demanded that a variety of video compression techniques and standards be developed. The major international standards recommended by ISO and the ITU make use of common video coding methods. The generalized digital video encoder introduced in Section 2 illustrates the spatial and temporal video compression elements that are central to the current MPEG-1, MPEG-2/H.262, H.261, and H.263 standards that have been developed over the past decade. They address a vast landscape of application requirements, from low- to high-bitrate environments, as well as stored video and multimedia to real-time videoconferencing and high-quality broadcast television.

The near future will drive video compression systems to incorporate support for more interactive functions, with the ability to define and download new functions to the encoder. New encoding methods currently being explored by the MPEG-4 and MPEG-7 standards look toward object-based encoding in which the encoder is not required to follow the international video transmission signal formats. These object-based techniques are expected to produce significant improvements in both encoder efficiency and functionality for the end user.

References

[1] ISO/IEC 11172 Information Technology: Coding of moving pictures and associated audio for digital storage media at up to ~1.5 Mbit/s, 1993.

[2] ISO/IEC JTC1/SC29/WG11, CD 13818: Generic coding of moving pictures and associated audio, 1993.

[3] CCITT Recommendation H.261: "Video Codec for Audio Visual Services at $p \times 64$ kbits/s," COM XV-R 37-E, 1990.

[4] H. Hseuh-Ming, and J. W. Woods, *Handbook of Visual Communications* (Academic, San Diego, CA, 1995), Chap. 6.

[5] J. W. Woods, *Subband Image Coding* (Kluwer, Norwell, MA, 1991).

[6] L. Wang and M. Goldberg, "Progressive image transmission using vector quantization on images in pyramid form," *IEEE Trans. Commun.*, **42(6)**, 1339–1349 (1989).

[7] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.* **27**, 379–423, 623–656 (1948).

[8] D. Huffman, "A method for the construction of minimum redundancy codes," *Proc. IRE* **40**, 1098–1101 (1952).

[9] P. W. Jones and M. Rabbani, *Digital Image Compression Techniques*, (SPIE, Bellingham, WA, 1991), p. 60.

[10] N. Ahmed, T. R. Natarajan, and K. R. Rao, "On image processing and a discrete cosine transform," *IEEE Trans. Comput.* **IT-23**, 90–93 (1974).

[11] J. J. Hwang and K. R. Rao, *Techniques and Standards For Image, Video, and Audio Coding* (Prentice-Hall, Upper Saddle River, NJ, 1996), p. 22.

[12] R. M. Gray, "Vector quantization," *IEEE ASSP Mag.* **IT-1**, 4–29 (1984).

[13] W. H. Equitz, "A new vector quantization clustering algorithm," *IEEE Trans. Acoust. Speech Signal Process.* **37**, 1568–1575 (1989).

[14] B. G. Haskell and J. O. Limb, "Predictive video encoding using measured subjective velocity," U.S. Patent No. 3,632,865, January, 1972.

[15] A. N. Netravali and J. D. Robbins, "Motion-compensated television coding: Part I," *Bell Syst. Tech. J.* **58**, 631–670 (1979).

[16] J. R. Jain and A. K. Jain, "Displacement measurement and its application in interframe image coding," *IEEE Trans. Commun.* **COM-29**, 1799–1808 (1981).

[17] T. Koga, "Motion compensated interframe coding for video conferencing," presented at the National Telecommunications Conference, New Orleans, LA, November, 1981.

[18] ITU-T SG 15 WP 15/1, Draft Recommendation H.263 (Video coding for low bitrate communications), Document LBC-95-251, Oct., 1995.

[19] M. Roser, "Extrapolation of a MPEG-1 video-coding scheme for low-it-rate applications," in *Video Communications and PACS for Medical Applications Proc. SPIE* 1977, 180–187 (1993).

[20] CCITT SG XV WP/1/Q4 Specialist Group on Coding for Visual Telephony, Description of Ref. Model 6 (RM6), Document 396, Oct. 1988.

[21] CCITT SG XV WP/1/Q4 Specialist Group on Coding for Visual Telephony, Description of Ref. Model 8 (RM8), Document 525, June, 1989.

6.2

Spatiotemporal Subband/ Wavelet Video Compression

John W. Woods,
Soo-Chul Han,
Shih-Ta Hsiang,
and T. Naveen
Rensselaer Polytechnic Institute

1 Introduction

This chapter is devoted to subband and wavelet video compression. We start out by showing the unity between these two approaches. They will be revealed to be essentially the same for digital video; hence our chapter title of subband/wavelet compression. Thus our chapter can be viewed as a companion to earlier chapters on wavelets (Chapter 4.1) and wavelet image compression (Chapter 5.4). We review image and video compression basics from the standpoint of subbands and wavelets. We treat subband/wavelet video compression itself in the next section, including the hybrid or recursive as well as nonrecursive methods that use a subband/wavelet transformation in the temporal direction also. There is the possibility of improving compression efficiency by performing the temporal filtering along the motion trajectory, if motion estimation is employed. In both cases efficiency can be improved by coding across the scales or subband levels by introducing a special zero symbol and forming a zero-tree structure.

For various reasons, initially related to compression efficiency, an object-based approach has been pursued. This means that the video is treated as being made up from separate objects moving and deforming in time. The main advantages of object-based coding have turned out to be in the areas of providing additional functionalities, such as object-based scalability and compression capability for composited images and videos. We present an object-based version of spatiotemporal subband/wavelet coding. Then we briefly present the topic of invertible motion compensated spatiotemporal coding. Here, even in the presence of half-pixel motion compensation, the synthesis operation can reconstruct the exact source video in the absence of quantization errors. These two techniques could be combined to achieve invertible subband/wavelet coding of spatiotemporal objects.

Currently with this writing, the JPEG 2000 standards body is adopting a subband/wavelet method for *image* coding. However, existent and emerging *video* compression standards are based on block processing using the discrete cosine transform (DCT) as the decorrelating transformation, followed by quantization and variable length coding. In this chapter we will review various methods for replacing the DCT by more general subband/wavelet transformations, both in hybrid coding employing spatial subbands, and in 3-D (spatiotemporal) subbands.

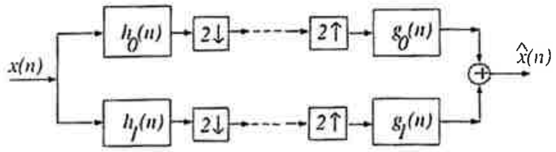


FIGURE 1 1-D subband/wavelet analysis and synthesis filter bank.

1.1 Subbands and Wavelets Reviewed

Subband methods started from work in digital signal processing in the area of speech compression. A major step was the invention of quadrature mirror filters (QMFs) by Esteban and Galland [1] in 1977. A very often used set of QMF filters appeared in Johnston's 1980 paper [2]. These filters, when applied in a 2-D separable manner, were found to be good for image coding too [3]. We summarize some results below. More details on subband/wavelet filters can be found in [4].

1.2 Subband/Wavelet Filter Sets

In Fig. 1, neglecting the coding errors and transmission losses, we can write

$$\begin{aligned}\hat{X}(\omega) = {} & \tfrac{1}{2}\,[G_0(\omega)H_0(\omega) + G_1(\omega)H_1(\omega)]\;X(\omega)\\ & + \tfrac{1}{2}\,[G_0(\omega)H_0(\omega+\pi)\\ & + G_1(\omega)H_1(\omega+\pi)]X(\omega+\pi).\end{aligned} \tag{1}$$

or equivalently in the Z transform domain:

$$\hat{X}(z) = \tfrac{1}{2}\begin{bmatrix} G_0(z)\\ G_1(z)\end{bmatrix}^T \begin{bmatrix} H_0(z)H_0(-z)\\ H_1(z)H_1(-z)\end{bmatrix}\begin{bmatrix} X(z)\\ X(-z)\end{bmatrix}.$$

A common goal is to design this analysis/synthesis filter bank to have the perfect reconstruction (PR) property, i.e., $\hat{X}(\omega) = X(\omega)$.

The second term in Eq. (1) is due to aliasing, which can be made to disappear (necessary and sufficient) by setting

$$G_0(\omega)H_0(\omega+\pi) + G_1(\omega)H_1(\omega+\pi) = 0. \tag{2}$$

The necessary and sufficient solution to Eq. (2) in the Z-transform domain is

$$\begin{bmatrix} G_0(z)\\ G_1(z)\end{bmatrix} = C(z)\begin{bmatrix} H_1(-z)\\ -H_0(-z)\end{bmatrix} \tag{3}$$

for some $C(z)$, which is usually taken to be a constant c. In the Fourier domain, this is then equivalent to

$$\begin{aligned} G_0(\omega) &= cH_1(\omega+\pi),\\ G_1(\omega) &= -cH_0(\omega+\pi),\end{aligned} \tag{4}$$

and in the spatial (time) domain,

$$\begin{aligned} g_0(n) &= c(-1)^n h_1(n),\\ g_1(n) &= -c(-1)^n h_0(n).\end{aligned} \tag{5}$$

Upon cancellation of the aliased component in the output, the overall transfer function is given by

$$\begin{aligned} T(\omega) &= \frac{\hat{X}(\omega)}{X(\omega)} = \frac{c}{2}[H_0(\omega)H_1(\omega+\pi) - H_0(\omega+\pi)H_1(\omega)],\\ T(z) &= \frac{\hat{X}(z)}{X(z)} = \frac{c}{2}[H_0(z)H_1(-z) - H_0(-z)H_1(z)].\end{aligned} \tag{6}$$

Ideally the filter bank output should be a delayed replica of the input:

$$T(\omega) = e^{-j\omega D}. \tag{7}$$

The necessary and sufficient condition for this is [5]

$$c/2[H_0(z)H_1(-z) - H_0(-z)H_1(z)] = \text{const } z^{-2l-1}, \quad l \in \mathcal{Z}, \tag{8}$$

where $\mathcal{Z}$ denotes the set of integers.

Here, we summarize some design considerations for subband/wavelet filters, some of which are conflicting. For image and video coding, these criteria need not be satisfied exactly and approximations are sufficient.

1. Easy to implement (computationally efficient). This could be achieved through one or more: symmetric filter coefficients, short-length filters, multiplierless implementation of the convolution, and fast transform equivalent of the convolution.
2. The wavelet basis functions generated by h_0 are orthogonal. That is, the impulse response of filter h_0 and its shifted versions (by even shifts) form an orthogonal set. This ensures that there is no redundancy in the transform coefficients.
3. For the same reason as above, the wavelet basis functions generated by h_0 and h_1 should be orthogonal.
4. PR holds in the absence of coding and channel errors.
5. The aliased components in the subbands should be small. This is because the upper subband may have to be truncated because of bit-rate constraints. It is achieved by making the frequency response of h_0 as close as possible to that of an ideal half-band filter.
6. The filters should have linear phase, which is important in image compression.
7. The overall transfer function $T(\omega)$ should be maximally flat at zero frequency. This is important because the energy in images is concentrated near zero frequency, and it is undesirable to introduce much distortion there.
8. The coding gain should be maximized. This would involve signal adaptive design of the filters.
9. The filters should be such that the energy of the signal is concentrated in a single subband (as much as possible). This criterion is necessary for coding efficiency.
10. Step response of the filters should have small overshoots. Otherwise, ringing artifacts occur in the encoded image.
11. Regularity: Iterated synthesis applied to a sequence consisting of only one nonzero entry should look reasonably

nice, even after several iterations. This feature is needed when one subband is made zero while encoding at low bit rate.

12. For optimality, the subband signals should be uncorrelated. That is, for a zero-mean wide-sense stationary (WSS) input $x(n)$,

$$E\{x_i(k)x_j(l)\} = \sigma_i^2\delta_{ij}\delta_{kl}\,\forall k,\, l, \quad i,\, j \in \{0, 1\}. \tag{9}$$

This would let us encode various subbands and their components independently, in the Gaussian case.

We say a subband filter set is orthogonal if criteria 2 and 3 are satisfied. We note that considerations 5 and 10 conflict. Though one cannot achieve zero overshoot and also an ideal frequency response, one can use a cost function for the optimization, which is a combination of both the step and frequency responses of the filters. Numerous approaches have been reported in the literature to design filters h_i and g_i, $i = 0, 1$, that satisfy exactly or approximately Eqs. (5) and (7), in addition to some of the other considerations given above. The QMF filters have the property that the high-pass and low-pass filters are mirror symmetric about $\omega = \pi/2$, but with only approximate perfect reconstruction. The biorthogonal case is a generalization of the PR orthogonal design wherein separate orthogonal basis filters can be used for analysis and synthesis. In such a case the h_i and g_i are less constrained than the orthogonal case in Eq. (6). It has been claimed that this extra freedom can result in significant improvement in coding efficiency. Biorthogonal filter design is considered in [6, 7], and the now widely used wavelet 9/7 biorthogonal filter set was first used for image coding in [6]. Both orthogonal and biorthogonal PR filter banks having the regularity property are related to wavelet theory (cf. Chapter 4.1). A wavelet transform splits the signal space in two, and then recursively splits the lower frequency half space in two, and so on. This is done for images by separable filtering, as mentioned above; i.e., filter the rows first and then the columns (cf. Chapter 5.4). For the video extension, one can continue this separable approach by addition of temporal domain filtering to accomplish an overall 3-D or spatiotemporal subband/wavelet transformation.

1.3 Optimal Subband/Wavelet Filters

Kronander [8] designed a linear phase biorthogonal filter, using a combination of step-response and frequency-response errors as the objective function.

References [9, 10] design a paraunitary filter bank that optimizes the coding gain for a given input signal. Assuming a constant quantizer performance factor [11], the coding gain over pulse code modulation (*PCM*) of a two-band subband scheme with an orthogonal filter bank is given by

$$G_{SBC} = \frac{\frac{1}{2}\left(\sigma_{x_0}^2 + \sigma_{x_1}^2\right)}{\left(\sigma_{x_0}^2\sigma_{x_1}^2\right)^{\frac{1}{2}}}, \tag{10}$$

where $\sigma_{x_0}^2$ and $\sigma_{x_1}^2$ are the variances of subband signals $x_0(n)$ and $x_1(n)$, respectively. Maximizing this gain involves finding the filter set that minimizes the variance of one of the two subbands. When the spectrum of the input signal $x(n)$ is nonincreasing and has components beyond $\omega = \pi/2$, which is true for many natural images, the design goal would be to approximate ideal half-band filters. A definitive treatment of this approach to optimal orthonormal coders is given by Vaidyanathan [12], where it is argued optimal biorthogonal coders cannot beat the performance of orthonormal coders if the power spectrum of the signal is flat over the subbands. Signal adapted finite-order filter design has been presented by Moulin *et al.* [13]. Again, by means of separable or row-column processing, this method can be extended to images and then onto image sequences or video.

1.4 Comparison of Two Subband/Wavelet Filter Sets

The power of filter sets for compression depends, of course, on the nature of the frequency decomposition as well as the nature of the filter. For this reason, it is of interest to compare the peak signal-to-noise ratio (PSNR) performance of some of these filters on a standard 10 subband wavelet decomposition versus a 16 subband full decomposition. The latter non-wavelet decomposition is sometimes called the 'wavelet packet' case. Some authors have found better PSNR performance of the full band case [14] over the wavelet, sometimes called dyadic or octave band, decomposition.

Knowing of the variety of filters that are available, there arises the question of how these various filters work in a coding context. Here we report on our test for the *Lena* image only and for two popular filters, the biorthogonal Daubechies 9/7 set [6] and from the oldest QMF design, we select Johnston's 16B [15]. The 16B is one of the first QMF filters and has been used for both audio and image coding from the early times. The more recent 9/7 filter has come from wavelet theory and is generally regarded as the best nonadapted filter to use currently for image compression. Both filters are used in a dyadic or octave band decomposition as well as a full or complete tree decomposition. The octave band decomposition is for three levels resulting in a traditional wavelet decomposition with 10 subbands. The full decomposition is for two levels and results in 16 subbands. A more thorough study of the effects of using different filters has been done by Villasenor [16]. Many notable individual coding results are posted at the website www.icsl.ucla.edu/~ipl/psnr_results.html.

We look at the *Lena* image and just two filters. The coder used is a one-class version of subband finite-state scalar quantization (SB-FSSQ) described in [17], and so does not implement prediction across the subbands or scales. The PSNR results are contained in Table 1, but can be summarized as follows. First the 16-band or nonwavelet decomposition is best at or above 0.5 bits/pixel, i.e., at higher qualities. At lower rates the 10-band octave or traditional wavelet decomposition works better. Having said this, we note that the PSNR difference between the

TABLE 1 PSNR comparison of Daubechies 9/7 vs. Johnston 16B filter sets on the *Lena* image

Daub 9/7				Johnston 16B			
10-band		16-band		10-band		16-band	
bpp	PSNR	bpp	PSNR	bpp	PSNR	bpp	PSNR
0.96	39.2	1.00	39.5	1.00	38.6	1.01	39.4
0.74	38.0	0.76	38.2	0.73	37.2	0.74	38.0
0.49	36.2	0.50	36.3	0.50	35.8	0.50	36.2
0.25	33.2	0.24	32.9	0.25	33.0	0.24	32.9
0.13	30.4	0.12	29.5	0.12	30.2	0.13	29.7

four cases at any given bit rate is never more than 1 dB and often much less. For example at 0.5 bpp, a 16-subband Johnston filter decomposition results in a PSNR of 36.2 dB, while the Daubechies 9/7 results in 36.3 dB, only a 0.1-dB difference. Both filter results are better for the full subband decomposition than for the wavelet decomposition. If we use a full 16-subband tree, the maximum observed difference is 0.2 dB. Visual differences are not judged as significant.

2 Video Compression Basics

Here we review some video compression basics relevant to spatiotemporal coding. We look at motion estimation and compensation first. This is followed by the transformation and quantization. Then we introduce the issue of scalability, which has been an interesting research topic, as well as being of concern to international standards bodies. The scalability properties of the spatiotemporal or 3-D filtering approaches has been considered one of their foremost advantages.

2.1 Motion Compensation

The motion estimation problem (cf. Chapter 6.1) for spatiotemporal subband/wavelet coding is somewhat different than that of hybrid coding. This is because in the scalable case, which is the main focus of the spatiotemporal coding, the lower frame-rate sequences are *created* by the motion compensated spatiotemporal filtering. Thus this is the ideal low frame-rate image sequence being communicated to the receiver. Any artifacts created by motion field errors will be seen directly in the lower frame rate output.

2.2 Transformation and Quantization

The role of the transformation is generally to reduce the dependence between the video samples. For example, linear transformations such as DCT and subband/wavelet filter trees and banks are known to reduce the correlation between transformed samples. In the Gaussian case, correlation and dependence are synonymous. More generally correlation and dependence typically reduce together, though this is not always the case. It is important to note that the transformation does not reduce entropy. A scalar or vector quantizer is then called on to provide the desired data compression. While the optimal quantizer (from a mean-square error viewpoint) will have the best performance, most modern coders use uniform step-size scalar quantizers, with a central dead zone to reject noise in the signal subband.

2.3 Scalabilities

Many applications of video coding require some sort of *scalability*, that is, the ability to usefully decode from only portions of the full compressed file. That is to say, we want one scalable coded file, consisting of a telescoping set of embedded files, that offers increasingly greater spatial resolution, higher frame rates, or a better signal-to-noise ratio (SNR). One motivation for scalability is for multicast on a heterogeneous computer network. If a certain part of the net contains only low-resolution terminals, then only that part of the scalable bit stream has to propagate there. Some receiving computers of varying clock speeds will only be able to keep up with lower resolution or lower frame-rate parts of the transmitted signal. Then an SNR scalable coder and appropriate decoder software will permit them all to get a usable image and keep up with the transmission. Alternatively, for a resolution or frame-rate scalable coder, we avoid the bandwidth inefficiency of the *ad hoc* solution of dropping frames at the receiver.

3 Subband/Wavelet Compression

There are basically two types of subband/wavelet video compression. One makes use of a frame-difference coder for the temporal direction amounting to a temporal differential PCM (DPCM) loop. Such a coder is called a *hybrid coder* when coupled with either a block transform or a subband/wavelet based coder in the spatial dimension. The other option is to use subband/wavelet coding for the temporal dimension too. Before presenting this 3-D or spatiotemporal subband/wavelet coder, we pause to look at the hybrid coder briefly.

3.1 Hybrid Subband/Wavelet Coder

This is motion compensated predictive coding with subband/wavelet filters used for the transformation instead of the DCT used in a standards-based coder like MPEG. Most subband/wavelet video coders are hybrid coders too. The class of hybrid coders is characterized by a very efficient one-frame recursive structure. While very efficient for implementation, and limiting the need for motion compensation to a frame-by-frame basis, this recursive structure can be a problem with regard to error propagation, scalability, picture in fastforward, and optimization of the coder. The latter arises because of the dependent frame nature of the hybrid coder's recursive structure.

3.2 Spatiotemporal Subband/Wavelet Coder

This type of coder is a spatiotemporal or 3-D subband/wavelet transformation, in contrast to the hybrid coder, which uses subband/wavelet filters only for the spatial transformation. There are two versions of these spatiotemporal subband/wavelet coders currently of interest; they differ in their use of motion compensation.

3.2.1 Without Motion Compensation

This is the simplest type of spatiotemporal subband/wavelet coder. The advantages of no motion compensation are

- computational simplicity,
- freedom from motion artifacts,
- easier transmission error concealment, and
- limited error propagation.

A number of such 3-D subband coders have been advanced in the literature [18–20]. Some have claimed to offer pictures equivalent to those of MPEG2 at similar rates [18, 21]. This is remarkable since no motion compensation is used. Of course the performance will vary with the motion content in the scene and whether the motion estimator is able to track it or not. For trackable moderate to high motion, we believe that motion compensation is still the best approach. Without motion compensation, the lower temporal video subbands will be blurred (or worse display multiple images or ghosts) when there is much motion. This is a serious disadvantage for scalable frame-rate coding.

3.2.2 With Motion Compensation

If we can afford to use motion compensation in our video coder, then we gain the added efficiencies of this method. There are two variants, the simpler of which uses just one global motion vector, which is suitable for camera-pan compensation [22]. Studies have indicated that camera pan constitutes a large portion of the motion seen in entertainment television. The next step is to get a fixed-size block-based motion estimate and compensation. More computation even can yield a variable block size and more accurate motion field [23–25], or still denser near-continuous motion fields. These latter two more accurate motion fields are important for spatiotemporal scalable methods, in which the motion compensated filter is used to generate the low frame-rate videos, which are the ideal videos that will be subject to the subsequent coding. Any motion compensation artifacts in these lower temporal subbands will inevitably show up in the received and decoded lower frame-rate videos.

3.3 Zero Coding and Embedding

Often, especially when high compression ratios are needed, the quantizer step size for the high-frequency subbands is large. Because of the central dead band of the normally used scalar quantizer, this makes its zero output value quite probable. Zero coding is a way to take advantage of this fact by attempting to code clusters or runs of these zero values together. This is done in MPEG by coding the run lengths in a so-called zigzag scan of the DCT coefficients. In subband/wavelet image coders, not only zero runs but zero clusters have been efficiently coded, most notably in the zero-tree image coder of Shapiro [26], who codes the zeros across spatial scales by introducing a special symbol called the *zero-tree root* for the often occurring situation in which a quantizer zero at one scale is associated with zero values at all finer spatial scales. This coder has been improved by Said and Pearlman in their set partitioning in hierarchical trees (SPIHT) [27], which processes lists of symbols related to significant and insignificant sets of wavelet coefficients. This image coder has been extended to video as 3-D SPIHT in [21]. These coders are made *embedded* by coding bit planes of coefficients in a most-significant-bit-first strategy, which results in a coded bit stream in which one can stop decoding after each bit plane and get the image or video represented to that level of significance. Thus this embedded property facilitates the SNR type of scalability that is desirable when compression is done once for many possible decodings at various quality levels, such as the computationally limited PC mentioned above. Resolution and frame-rate scalability were not addressed in these papers. Interestingly, the four class SB-FSSQ coder [17] has better PSNR than the embedded zero-tree wavelet (EZW) coder [26] on the *Lena* image.

4 Object-Based Subband/Wavelet Compression

At least two problems have prevented object-based video coding systems from outperforming standard block-based techniques. Object segmentation is a very difficult problem because of its sensitivity and complexity. Also, we have the additional need to transmit the contour or shape of the object, leading to additional bit rate. So, the gain in coding the objects must outweigh the need to transmit the additional contour information.

An object-based coder addressing these issues was presented in [28]. The extraction of the moving objects is performed by a joint motion estimation and segmentation algorithm based on Markov random field (MRF) models (cf. Chapter 4.2). In this approach, the object motion and shape are guided by the spatial color intensity information, thus utilizing the observation that in an image sequence, motion and intensity boundaries usually coincide. This not only improves the motion estimation/segmentation process itself in extracting meaningful objects true to the scene, but it also aids the process of coding the object intensities because a given object has a certain spatial cohesiveness. The MRF formulation also allows temporal linking of the objects, thus creating *spatiotemporal objects*. This helps stabilize the object segmentation process in time, and more importantly for coding, allows the object boundaries to be predicted temporally by using the motion information. An efficient temporal updating scheme to encode the object boundaries results in a significant reduction in bit rate while preserving the accuracy of the boundaries. With the linked objects, uncovered regions can be deduced in a systematic fashion. New objects are detected by utilizing both the motion and intensity information. The interiors of the objects are encoded adaptively, meaning that objects well described by the motion parameters are encoded in an "inter" mode, while those that cannot be predicted in time are encoded in an "intra" mode. This is analogous to P blocks and I blocks in the MPEG coding structure (cf. Chapters 6.4 and 6.5), where we now have P objects and I objects. I-object coding is feasible because the object segments are based on intensity information. The subband/wavelet approach [29] is adopted in spatial coding the objects. Both hybrid [28] and spatiotemporal [30] versions of this object-based subband/wavelet coder were developed.

4.1 Joint Motion Estimation and Segmentation

The main objective is to segment the video scene into objects that are undergoing distinct motion and to find the parameters that describe the motion. We have adopted a Bayesian formulation based on an MRF model to solve this challenging problem. The MRF approach was initially used in motion segmentation and motion estimation in separate works. Because of the interdependency of the two problems, algorithms to perform the motion estimation and segmentation jointly have been proposed [31,32].

4.1.1 Problem Formulation

Let $\mathbf{I}^t$ represent the frame at time t of the discretized image sequence. The motion field $\mathbf{d}^t$ represents the displacement between $\mathbf{I}^t$ and $\mathbf{I}^{t-1}$ for each pixel. The segmentation field $\mathbf{z}^t$ consists of numerical labels at every pixel, with each label representing one moving object, i.e., $\mathbf{z}^t(\mathbf{x}) = n$ $(n = 1, 2, \ldots, N)$, for each pixel location $\mathbf{x}$ on the lattice Λ. Here, N refers to the total number of moving objects. With the use of this notation, the goal of motion estimation/segmentation is to find $\{\mathbf{d}^t, \mathbf{z}^t\}$ given $\mathbf{I}^t$ and $\mathbf{I}^{t-1}$. We further assume that $\mathbf{d}^{t-1}$ and $\mathbf{z}^{t-1}$ are available, making it possible to impose temporal constraints and to link the object labels.

We adopt the maximum *a posteriori* (MAP) formulation to provide estimates $\hat{\mathbf{d}}^t, \hat{\mathbf{z}}^t$ by maximizing the joint conditional density $p(\mathbf{d}^t, \mathbf{z}^t \mid \mathbf{I}^t, \mathbf{I}^{t-1})$. With the use of Bayes' rule, this is simplified to the equivalent maximization of the product of mixed conditional densities and probabilities:

$$p(\mathbf{I}^{t-1} \mid \mathbf{d}^t, \mathbf{z}^t, \mathbf{I}^t)\, p(\mathbf{d}^t|\mathbf{z}^t, \mathbf{I}^t)\, P(\mathbf{z}^t \mid \mathbf{I}^t), \qquad (11)$$

each of which will be explained in the paragraphs that follow, where we incorporate various assumptions and models about our motion and segmentation field in formulating these terms.

4.1.2 Probability Models

The first term of Eq. (11) is the likelihood functional that describes how well the observed images match the motion field data. It reflects the relationship between the gray-level changes between frame $t-1$ and t that are corrupted by additive noise. Thus, the actual observed image $\mathbf{I}^t$ is regarded as a noisy version of the original image $\mathbf{G}^t$, or $\mathbf{I}^t(\mathbf{x}) = \mathbf{G}^t(\mathbf{x}) + \eta(\mathbf{x})$. Ignoring such factors as illumination changes, we assume the change of gray level between the two frames to be only due to object motion, and we have $\mathbf{G}^t(\mathbf{x}) = \mathbf{G}^{t-1}(\mathbf{x} - \mathbf{d}(\mathbf{x}))$. If the noise is assumed to be white Gaussian with zero mean and variance σ^2, $p(\mathbf{I}^{t-1} \mid \mathbf{d}^t, \mathbf{z}^t, \mathbf{I}^t)$ is also Gaussian with $p(\mathbf{I}^{t-1} \mid \mathbf{d}^t, \mathbf{z}^t, \mathbf{I}^t) = Q_l^{-1} \exp\{-U_l(\mathbf{I}^{t-1} \mid \mathbf{d}^t, \mathbf{I}^t)\}$ where the energy function $U_l(\mathbf{I}^{t-1} \mid \mathbf{d}^t, \mathbf{I}^t) = \sum_{\mathbf{x} \in \Lambda} (\mathbf{I}^t(\mathbf{x}) - \mathbf{I}^{t-1}(\mathbf{x} - \mathbf{d}^t(\mathbf{x})))^2/2^2$ and Q_l is a normalization constant.

The second term of Eq. (11) is the *a priori* density of motion $p(\mathbf{d}^t \mid \mathbf{z}^t, \mathbf{I}^t)$ and thus enforces prior constraints on the motion field. We adopted a coupled MRF model in [28] to govern the interaction between the motion field and segmentation field both spatially and temporally. The probability density and corresponding energy function is given as $p(\mathbf{d}^t \mid \mathbf{z}^t, \mathbf{I}^t) = Q_d^{-1} \exp\{-U_d(\mathbf{d}^t \mid \mathbf{z}^t)\}$ and

$$\begin{aligned} U_d(\mathbf{d}^t \mid \mathbf{z}^t) = {} & \lambda_1 \sum_{\mathbf{x}} \sum_{\mathbf{y} \in \mathcal{N}_\mathbf{x}} \|\mathbf{d}^t(\mathbf{x}) - \mathbf{d}^t(\mathbf{y})\|^2 \delta(z^t(\mathbf{x}) - z^t(\mathbf{y})) \\ & + \lambda_2 \sum_{\mathbf{x}} \|\mathbf{d}^t(\mathbf{x}) - \mathbf{d}^{t-1}(\mathbf{x} - \mathbf{d}^t(\mathbf{x}))\|^2 \\ & - \lambda_3 \sum_{\mathbf{x}} \delta(z^t(\mathbf{x}) - z^{t-1}(\mathbf{x} - \mathbf{d}^t(\mathbf{x}))). \end{aligned} \qquad (12)$$

where refers to the usual Kronecker delta function[1], $\|\cdot\|$ is Euclidean norm in $\mathbf{R}^2$, and $\mathcal{N}_\mathbf{x}$ indicates a small neighborhood of $\mathbf{x}$. The first term encourages motion vectors be locally smooth, but only *within* each object. The second term links the motion vectors along the motion trajectory. The last term encourages the object labels to be consistent along the motion trajectories.

Now as to the third term on the right-hand side of Eq. (11), $P(\mathbf{z}^t \mid \mathbf{I}^t)$, it models our *a priori* expectations for the object label field itself. In the temporal direction, we have already modeled the object labels to be consistent along the motion trajectories. Our model incorporates the spatial intensity information ($\mathbf{I}^t$) based on the reasonable assumption that object discontinuities coincide with spatial intensity boundaries. The segmentation field is a discrete-valued MRF, $P(\mathbf{z}^t \mid \mathbf{I}^t) = Q_z^{-1} \exp\{-U_z(\mathbf{z}^t \mid \mathbf{I}^t)\}$ with the energy function given as $U_z(\mathbf{z}^t \mid \mathbf{I}^t) = \sum_\mathbf{x} \sum_{\mathbf{y} \in \mathcal{N}_\mathbf{x}} V_c(z(\mathbf{x}), z(\mathbf{y}) \mid \mathbf{I}^t)$, where

$$V_c(z(\mathbf{x}), z(\mathbf{y}) \mid \mathbf{I}^t) = \begin{cases} -\gamma & \text{if } z(\mathbf{x}) = z(\mathbf{y}),\ s(\mathbf{x}) = s(\mathbf{y}) \\ 0 & \text{if } z(\mathbf{x}) = z(\mathbf{y}),\ s(\mathbf{x}) \neq s(\mathbf{y}) \\ +\gamma & \text{if } z(\mathbf{x}) \neq z(\mathbf{y}),\ s(\mathbf{x}) = s(\mathbf{y}) \\ 0 & \text{if } z(\mathbf{x}) \neq z(\mathbf{y}),\ s(\mathbf{x}) \neq s(\mathbf{y}). \end{cases} \qquad (13)$$

Here, $\mathbf{s}$ refers to the spatial segmentation field that is predetermined from $\mathbf{I}$. As a simplification, we treat s as a *deterministic*

[1] The Kronecker delta function $\delta(\cdot)$ assigns the value $\delta = 1$ when its argument is 0 and $\delta = 0$ otherwise.

field that can be calculated uniquely from **I** alone. According to Eq. (13), if the spatial neighbors **x** and **y** belong to the same *intensity-based* object ($s(\mathbf{x}) = s(\mathbf{y})$), then the two pixels are encouraged to belong to the same *motion-based* object. This is achieved by the $\pm\gamma$ terms. In contrast, if **x** and **y** belong to *different* intensity-based objects ($s(\mathbf{x}) \neq s(\mathbf{y})$), we do not enforce z to be either way, and hence the 0 terms in Eq. (13). This slightly more complex model ensures that the moving object segments we extract have some sort of spatial cohesiveness as well. This is a very important property for our adaptive coding strategy, presented in the paragraphs that follow.

4.1.3 Maximation Approach

As a result of the equivalence of MRFs and *Gibbs densities*, i.e., those densities that can be written as the exponential of the negative of an energy function (cf. Chapter 4.2), the MAP solution amounts to a minimization of the sum of these energies. To ease the computation, a two-step iterative hierarchical procedure is implemented, in which the motion and segmentation fields are found in an alternating fashion, assuming the other is given. Mean field annealing is used for the motion field estimation, while the object label field is found by a deterministic iterated conditional modes (ICM) algorithm [33].

4.1.4 Video Object Segmentation Results

In Figs. 2 and 3, the segmentation results for *Miss America* are displayed in a horizontal versus time plot, corresponding to a fixed vertical position. We see that the segments generally follow the object in the scene and are coherent over time. We can see that the MRF model produced smooth vectors within the objects with definitive discontinuities at the intensity boundaries. Also, it can be observed that the object boundaries relate well to the "real" objects in the scene.

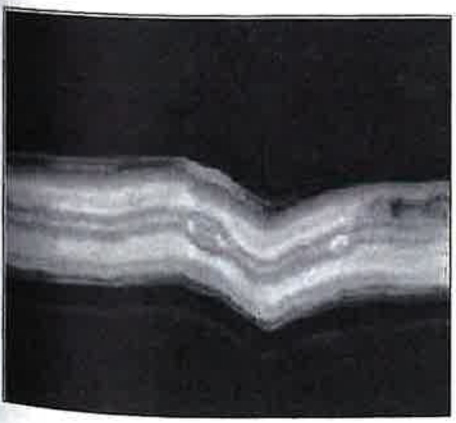


FIGURE 2 *Miss America* horizontal vs. time display. (Reprinted by permission from *Image and Video Compression*, P. Topiwala, ed., Kluwer Academic Publishers 1998.)

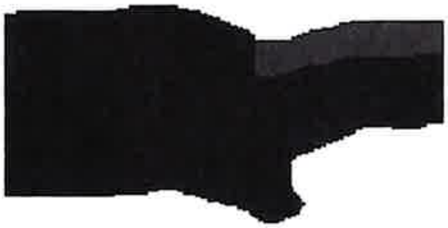


FIGURE 3 Segmented horizontal vs. time display. (Reprinted by permission from *Image and Video Compression*, P. Topiwala, ed., Kluwer Academic Publishers 1998.)

4.2 Coding of Video Objects

The coding of the object interior is performed by adaptive coding. Objects that can be described well by the motion were encoded by motion compensated predictive (MCP) coding in hybrid object-based (OB)-MCP [28], and those that cannot were encoded in the "intra" mode. The coding was done independently on each object, using spatial subband/wavelet coding. Since the objects are arbitrarily shaped, the efficient signal extension method proposed by Barnard [29] was applied.

Although the motion compensation was relatively good for most objects at most frames, the flexibility to switch to the intramode (I mode) in certain cases is inevitable. For instance, when a new object appears from outside the scene, it cannot be properly predicted from the previous frame. Thus, these new objects must be coded in the I mode. This includes the initial frame of the image sequence, where all the objects are considered new. Even for "continuing" objects, the motion might be too complex at certain frames for our model to describe properly, resulting in poor prediction. This is another case when objects should be encoded in the I mode. Such classification of objects into I objects and P objects is analogous to P blocks and I blocks in current MPEG video standards (cf. Chapters 6.4 and 6.5). Each of these linked spatiotemporal objects can also be coded by a 3-D spatiotemporal coder as in [30], offering scalability and robustness advantages over the hybrid OB-MCP method, and with, it turns out, almost the same performance.

4.2.1 Object Motion Field

The motion analysis provides us with the boundaries of the moving objects and a dense motion field within each object. An affine parametric representation can provide a smooth and efficient fit to each object's motion. Potential new objects can be found for regions where the fit fails. By modeling the motion of the temporally linked objects with affine parameters, one

TABLE 2 PSNR results for OB-3DSBC

Sequence	Bit Rate (kbps)	Channel	OB-3DSBC (PSNR)	H.263 (PSNR)
Miss America (15 fps)	20	Y	37.5	37.9
		U	38.9	38.5
		V	37.6	37.4
Carphone (15 fps)	40	Y	33.1	33.4
		U	38.3	38.6
		V	38.9	38.1

Source: From *Image and Video Compression*, P. Topiwala, ed., Kluwer Academic Publishers 1998.

reduces the bit rate to encode the object boundaries significantly [28,30]. Furthermore, one can extract uncovered regions simply by comparing the object location and motion parameters between two frames.

Because the objects are linked in time, covered/uncovered region extraction merely involves projecting the motion vectors in time and comparing labels. More specifically, for the uncovered regions in frame t to be found, each pixel is projected back to frame $t-1$ according to its synthesized motion vector. The uncovered pixels are simply those whose object labels don't match along the trajectory.

4.2.2 Coding the Object Boundaries

We have already seen that temporally linked objects in an object-based coding environment offer various advantages. However, the biggest advantage comes in reducing the contour information rate. Using the object boundaries from the previous frame and the affine transformation parameters, one can predict the boundaries with a good deal of accuracy. Some small error occurs near boundaries, and one can simply encode these by using 1-bit flags.

4.3 Object Motion/Segmentation Coding

The object-based 3-D subband/wavelet coding (OB-3DSBC) coder was tested on the QCIF resolution *Miss America* and *Carphone* sequences. Simulations were performed at the frame rate of 15 frames/s. The object segmentation and motion analysis from [28] was used. The target bit rate was 20 kbps at the full frame rate for *Miss America* and 40 kbps for *Carphone*, with the bits being divided equally among the group of pictures (GOPs) except for the first one. The first GOP was assigned twice as many bits as the other GOPs to account for the I-tLL band. For comparison, we obtained results at the same frame and bit rate with an H.263 standard coder (cf. Chapter 6.1). The average PSNRs are summarized in Table 2.

Figure 4 displays full-rate reconstruction results from the various methods for *Carphone*, with corresponding H.263 results shown in Fig. 5. In terms of the PSNR, we can see that the OB-3DSBC is somewhat worse (by 0.2–0.4 dB) than the H.263 coder. The OB-MCP coder results are slightly better in PSNR and are shown in [28]; however, the difference in visual quality with OB-3DSBC is minimal. On the plus side, the OB-3DSBC gives us a natural scalability option in frame rate, i.e., the flexibility of decoding the given bit stream at multiple frame-rates [30].

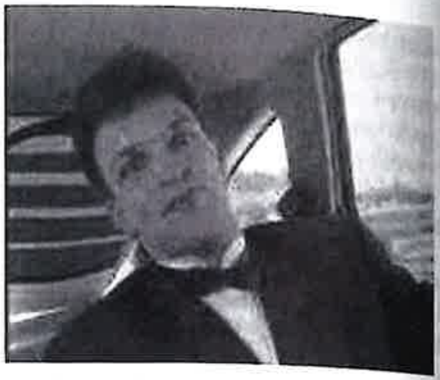


FIGURE 4 OB-3DSBC coder result. (Reprinted by permission from *Image and Video Compression*, P. Topiwala, ed., Kluwer Academic Publishers 1998.)

5 Invertible Subband/Wavelet Compression

The spatiotemporal coding presented in Section 3 has the problem of requiring interpolation to *create* the lower frame-rate videos. Even in the absence of any quantization error, the interpolation step will cause some distortion in the lower frame-rate videos. The result is that the above presented technique does not work that well for high quality (read high bit rates). To extend the technique to high quality and also high resolution, we need to address this problem. The interpolation is only needed when motion compensation is used at subpixel accuracy, but this is necessary for high-efficiency coding. Also, the motion compensation itself is a big cause of artifacts at the lower frame rates, where it is more inaccurate.

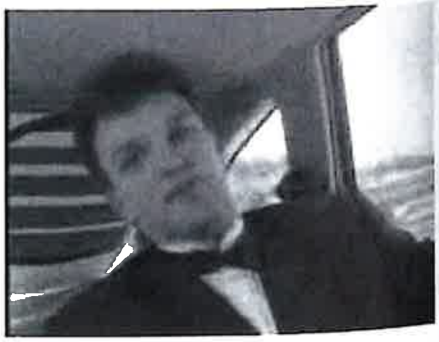


FIGURE 5 H.263 coder result. (Reprinted by permission from *Image and Video Compression*. P. Topiwala, ed., Kluwer Academic Publishers 1998.)

Because of its high-energy compaction and nonrecursive coding structure, spatiotemporal (3-D) subband/wavelet coding with motion compensation (MC-3DSBC) has been demonstrated to outperform conventional hybrid coders in compression efficiency [23, 25, 34] and in robustness for video transmission.

It is widely acknowledged that motion compensation with half-pixel accuracy is necessary in order to effectively reduce the energy of the displaced frame difference (DFD). Since the high-frequency output of the temporal two-tap analysis filter bank utilized in [34, 35] is the scaled difference of the previous and current frames, they adopted half-pixel accuracy for MC temporal filtering in order to reduce the energy of the high-frequency band. The images therein had to be interpolated at both analysis and synthesis stages, and the resulting systems were thus not invertible. Therefore, reconstruction error was introduced even without any coding distortion. This excluded the technique from high-quality video coding applications and also limited the number of analysis/synthesis stages allowed. In [25,34], two stages of temporal decomposition were applied in order to avoid buildup of reconstruction error from the analysis/synthesis system. For the HDTV application, only one stage could be used in [36]. To further enhance coding efficiency, the images of the lowest temporal band from the same GOP were encoded by temporal DPCM. Therefore, the overall system still could not fully avoid recursive coding structures and their disadvantages.

In [37], we presented an invertible 3-D or spatiotemporal subband/wavelet system with half-pixel-accurate motion compensation for video coding. We term it *invertible motion-compensated 3DSBC*, or IMC-3DSBC. There we looked at temporal decomposition of the progressively scanned image sequence as a kind of downconversion of the sampling lattice from the interlaced format to the progressive format, following the suggestion in [38]. We thus extended the method of [38], intended for interlaced/progressive scan conversion, to our analysis/synthesis system IMC-3DSBC. An important feature of the new system is that it guarantees perfect reconstruction while high-energy compaction is retained.

It is known that optimal bit allocation for conventional hybrid coders is very complex because of the frame-to-frame dependent quantization structure resulting from the DPCM coding loop [39]. In contrast, in a subband-based coder, coefficients of individual subbands are quantized and coded independently. Optimal bit allocation is therefore possible. However, since MC-DPCM was still used to encode frames of the lowest temporal band in the earlier MC-3DSBC [25, 34], bit allocation could not be fully optimized for the GOPs. In the new system, the input video is decomposed into four temporal stages without build-up of reconstruction error. The GOP consisting of 16 frames does not contain any dependent coding structure at all. Therefore, if the effects of side information are neglected, each GOP can be optimally encoded in an operational rate-distortion sense.

Figure 6 shows PSNR coding results versus bit rate of OB-3DSBC and MPEG-2 (TM5) for the color SIF version of the *Mobile Calendar* test clip. Note that the improvement of MC-3DSBC drops off, and will actually saturate, at the higher bit rates, while IMC-3DSBC does not. Notice the 2–3 dB improvement over MPEG-2, which is largely due to optimization, but which in turn is easier for nonrecursive coders.

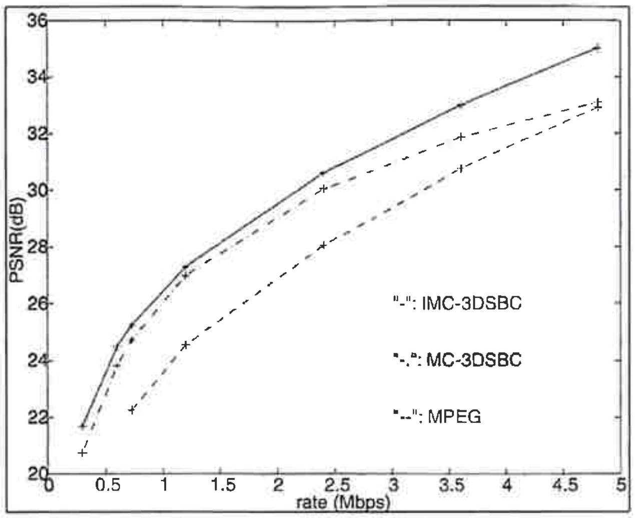


FIGURE 6 *Mobile Calendar* PSNR vs. bit rate for hybrid and spatiotemporal subband/wavelet object coders + MPEG2 (TM5).

6 Summary and Look Forward

This chapter has presented 3-D or spatiotemporal coding using subband/wavelet methods. We have first reviewed available filters and compared results. We related the spatiotemporal methods to hybrid methods such as MPEG and hybrid subband/wavelet. We have presented spatiotemporal coding for a low bit-rate, object-based coder, and we addressed the needs for higher rates and resultant quality by showing a method for invertible motion compensated spatiotemporal coding. We believe that future work should extend the invertible coder to code objects and at higher qualities and bit rates.

References

[1] D. Esteban and C. Galand, "Application of quadrature mirror filters to split band voice coding schemes," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing* (IEEE, New York, 1997), pp. 191–195.

[2] J. D. Johnston, "A filter family designed for use in quadrature mirror filter banks," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing* (IEEE, New York, 1980), pp. 291–294.

[3] J. W. Woods and S. D. O'Neil, "Sub-band coding of images," *IEEE Trans. Acoust. Speech Signal Process.* **ASSP-34**, 1278–1288 (1986).

[4] T. Naveen and J. Woods, "Subband and wavelet filters for high-definition video compression," in *Handbook of Visual Communications*, H.-M. Hang and J. Woods, eds. (Academic, New York, 1995).

[5] M. Vetterli and C. Herley, "Wavelets and filter banks: theory and design," *IEEE Trans. Signal Process.* **40**, 2207–2232 (1992).

[6] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," *IEEE Trans. Image Process.* **1**, 205–220 (1992).

[7] A. Cohen, I. Daubechies, and J.-C. Feauveau, "Biorthogonal bases of compactly supported wavelets," *Commun. Pure Appl. Math.* **45**, 485–560 (1992).

[8] T. Kronander, *Some Aspects of Perception Based Image Coding* (Linkoping U. Dissertations, Linkoping, Sweden, 1989).

[9] P. Desarte, B. Macq, and D. T. M. Slock, "Signal-adapted multiresolution transform for image coding," *IEEE Trans. Inf. Theory* **38**, 897–904 (1992).

[10] D. Taubman and A. Zakhor, "A multi-start algorithm for signal adaptive subband systems," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing* (IEEE, New York, 1992), Vol. III, pp. 213–216.

[11] N. S. Jayant and P. Noll, *Digital Coding of Waveforms* (Prentice-Hall, Englewood Cliffs, NJ, 1984).

[12] P. P. Vaidyanathan, "Theory of optimal orthonormal subband coders," *IEEE Trans. Signal Process.* **46**, 1528–1543 (1998).

[13] P. Moulin, M. Anitescu, K. O. Kortnek, and F. A. Potra, "The role of linear semi-infinite programming in signal-adapted qmf band design," *IEEE Trans. Signal Process.* **45**, 2160–2174 (1997).

[14] M. Vetterli and J. Kovacevic, *Wavelets and Subband Coding* (Prentice-Hall, Englewood Cliffs, NJ, 1995).

[15] J. D. Johnston, "A filter family designed for use in quadrature mirror filter banks," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing* (IEEE, New York, 1980), pp. 291–294.

[16] J. D. Villasenor, B. Belzer, and J. Liao, "Wavelet filter evaluation for image compression," *IEEE Trans. Image Process.* **2**, 1053–1060 (1995).

[17] T. Naveen and J. W. Woods, "Subband finite state scalar quantization," *IEEE Trans. Image Process.* **5**, 150–155 (1996).

[18] W. E. Glenn, J. Marcinka, and R. Dhein, "Simple scalable video compression using 3-D subband coding," *SMPTE J.* **Mar.**, **106**, 140–143 (1996).

[19] C. Podilchuk, N. Jayant, and N. Farvardin, "Three-dimensional subband coding of video," *IEEE Trans. Image Process.* **4**, 125–139 (1995).

[20] T. Meng, B. M. Gordon, E. K. Tsern, and A. C. Hung, "Portable video-on-demand in wireless communications," *Proc. IEEE* **83**, 659–680 (1995).

[21] W. A. Pearlman, B.-J. Kim, and Z. Xiong, "Embedded video coding with 3D SPIHT," in *Wavelet Image and Video Compression*, P. N. Topiwala, ed. (Kluwer, Boston, MA, 1998).

[22] D. Taubman and A. Zakhor, "Multirate 3-D subband coding of video," *IEEE Trans. Image Process.* **3**, 572–588 (1994).

[23] J. R. Ohm, "Three-dimensional subband coding with motion compensation," *IEEE Trans. Image Process.* **3**, 559–571 (1994).

[24] G. Lilienfield and J. W. Woods, "Scalable high-definition video coding," in *Proc. ICIP-95* (IEEE, Washington, DC) pp. 567–570.

[25] S.-J. Choi and J. W. Woods, "Motion-compensated 3-D subband coding of video," *IEEE Trans. Image Process.* **8**, 155–167 (1999).

[26] J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Trans. Signal Process.* **41**, 3445–62 (1993).

[27] A. Said and W. A. Pearlman, "A new fast/efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. Video Technol.* **6**, 243–250 (1996).

[28] S.-C. Han and J. W. Woods, "Adaptive coding of moving objects for very low bit-rates," *IEEE Trans. Select. Areas in Commun. Spec. Issue* **16**, 56–70 (1998).

[29] H. J. Barnard, "Image and video coding using a wavelet decomposition," Ph.D. dissertation (Delft U. of Technology, The Netherlands, 1994).

[30] S.-C. Han and J. W. Woods, "Scalable object-based 3-D subband/wavelet coding of video," submitted, 1998.

[31] C. Stiller, "Object-oriented video coding employing dense motion fields," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing* (IEEE, New York, 1994), Vol. V, pp. 273–276.

[32] M. Chang, I. Sezan, and A. Tekalp, "An algorithm for simultaneous motion estimation and scene segmentation," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing* (IEEE, New York, 1994), Vol. V, pp. 221–224.

[33] A. M. Tekalp, *Digital Video Processing* (Prentice-Hall, Upper Saddle River, NJ, 1995).

[34] S.-J. Choi and J. W. Woods, "Three-dimensional subband/wavelet coding of video with motion compensation," in *VCIP-97, Proc. SPIE* **3024-17**, 96–104 (1997).

[35] J. Ohm, "Three-dimensional subband coding with motion compensation," *IEEE Trans. Image Process.* **3**, 559–571 (1994).

[36] G. Lilienfield and J. W. Woods, "Scalable high definition video coding," in *VCIP-98, Proc. SPIE* **3309**, 158–169 (1998).

[37] S.-T. Hsiang and J. W. Woods, "Invertible three-dimensional analysis/synthesis system for video coding with half-pixel accurate motion compensation," in *VCIP-99, Proc. SPIE* **3653**, 537–546 (1999).

[38] J.-R. Ohm, "Variable-raster multiresolution video processing with motion compenstion techniques," in *ICIP-97, Proc. IEEE* (Santa Barbara, CA, 1997), **1** , pp. 759–.

[39] K. Ramchandran, A. Ortega, and M. Vetterli, "Bit allocation for dependent quantization with applications to multiresolution and mpeg video coders," *IEEE Trans. Image Process.* **3**, 533–545 (1994).

6.3

Object-Based Video Coding

Touradj Ebrahimi
and Murat Kunt
Swiss Federal Institute of Technology — EPFL

1 Introduction

Conventional digital image and image sequence coding has historically relied on a number of simple yet powerful concepts. An original image is converted into a digital format by sampling in space and time, and by quantizing in brightness or color. Messages defined by using this basic data format are referred to as being in "canonical form." Codewords have been assigned to messages in a variety of ways, motivated by the information theory framework. Examples of messages include pairs of adjacent pixels, groups of pixels within a geometrically simple data independent structure (e.g., a square image block), or a linear reversible transform of these pixels (such as the discrete cosine transform, or DCT). Statistical distributions of the messages have been used to determine optimal codeword assignments. The compression performance of these types of schemes saturated quickly. Natural images and image sequences are anything but stationary, meaning that the statistical properties of image data are variable over space and time. Although interesting, adaptive sampling is impractical. Furthermore, the entropy of a natural scene is hardly known and depends heavily, if not uniquely, on the model used to estimate image statistics and statistical dependencies. Last but not least, data independent structures such as Cartesian sampling grids (or square data blocks, as used in MPEG, for example) cannot describe nonstationarities and hence cannot serve as efficient data structures for images and image sequences.

Improvements have come by representing visual data in terms of regions, defined by their contour and texture, possibly corresponding to objects or to parts of objects. This approach closes the gap between technical systems and the human visual system (HVS), the latter usually being the last element of an image processing system. It also makes it possible to emphasize visually sensitive data while neglecting visually insignificant information. Of course, the raw data resulting from sampling and quantization must be transformed into this representation. Once the regions are obtained, there is still a challenging step to connect regions belonging to the same visual object. As a byproduct to compression and representation efficiency, this approach has paved the way to a number of new functionalities, such as interaction with regions and objects. This so-called second-generation concept is now widely accepted and has become the basic philosophy of the new MPEG-4 standard (Chapter 6.5).

Unfortunately, there is no single compression method or algorithm that can efficiently compress all possible image regions or objects, just as there is no single tool to repair a car. The ultimate representation is then to assign the tool to the information. Each type of visual information, region or object, can be compressed by the most efficient algorithm. The label of the algorithm is appended to the data and algorithms are accumulated in a tool box. This approach is called *dynamic coding.*

The chapter is organized as follows. In the next section, second-generation coding is presented as the basis for object-based coding. Section 3 describes an efficient and relatively simple way of encoding video information using objects. It has three main components based on the handling, respectively, of shape, motion, and texture. The components of the scheme are designed in such a way that each one allows progressive transmission or retrieval. Integrating these components into an overall progressive coding scheme is described in Section 4. The dynamic coding concept, together with a few illustrations, is developed in Section 5, before the conclusions in the last section.

2 Second-Generation Coding

The most widely used approach to represent still and moving pictures in the digital domain is based on pixels. This is mainly because pixel-based acquisition and reproduction of digital visual information are mature and relatively cheap technologies, as they produce uniformly sampled data. In parallel, we can view the lowest level of the HVS (rods and cones in the retina) as a non-uniformly sampled acquisition system [1, 2]. Compared to its technical counterpart, this system has, however, incredible complexity and sophistication in its higher levels. In a pixel-based representation, an image or a video is modeled as a set of pixels (with associated properties such as a given color or motion) the same way the physical world is made of atoms. Until recently, pixel-based image processing was the only digital representation available for the processing of visual information, and therefore the majority of techniques known today rely on this representation. It was in the mid-1980s that, for the first time, motivated by studies of the mechanisms of the human vision system, researchers developed other representation techniques [3]. The main idea behind this effort was that, since the HVS is in the majority of cases the final stage in the image processing chain, then a representation that matches the HVS will be more efficient in the design of image processing and coding systems. Non-pixel-based representation techniques for coding (also called second-generation coding) have been found to be superior in coding efficiency at very high compression ratios, when they are when compared with pixel-based representation methods [3].

Figure 1 depicts a representation pyramid illustrating various methods used to represent visual information and their relationships. Linear transform and (motion-compensated) predictive coding, which can be considered special cases of pixel-based representation techniques, have also shown outstanding results in compression efficiency for the coding of still images and video. One reason is that digital images and video are captured and therefore mainly available in a pixel-based form, as this is the only way we can acquire them today. In order to apply a non-pixel-based approach, either the input data should be captured in a non-pixel-based form, or the available pixel-based data have to be converted to a non-pixel-based representation, which brings additional complexity but also other inefficiencies. Examples of such conversions are depicted in Fig. 1 and can vary from simple visual primitive extraction methods to more sophisticated object segmentation and tracking techniques. An important class of non-pixel-based representation schemes is that of content-based representation. In this approach, an image is seen as a set of visual primitives (edges, contour, texture, etc.) containing the most salient visual information in the scene.

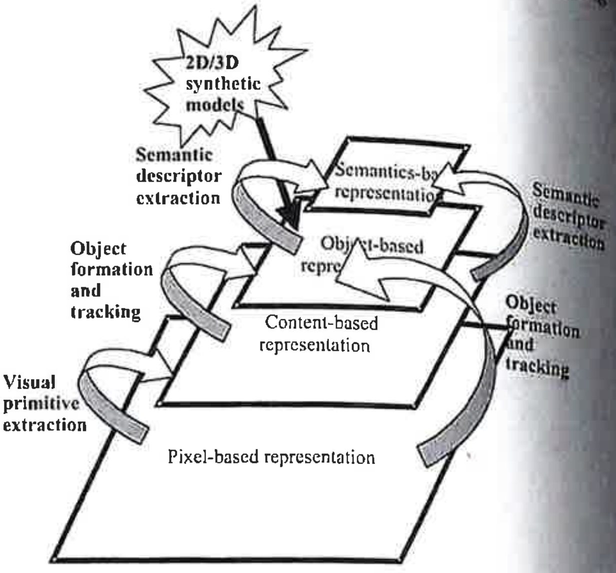


FIGURE 1 Visual information representation pyramid and its internal structure.

Among content-based representations, region-based and ultimately object-based visual data representations are very important classes. Here, regions are defined as segments in an image that share a common property, while objects are defined as sets of regions that represent a semantically meaningful entity in an image [4]. In object-based representations, objects replace pixels. An image or a video is seen as a set of objects that cannot be broken into smaller elements. In addition to texture (color) and motion properties, shape information is also needed in order to completely define any object. The shape in this case can be seen as a force field keeping together the elements of an image or video object like atoms in a molecule or a physical object. Once you grab a corner of an object, the rest comes with it because the force field has glued all atoms of the object together. The same is true in an object-based representation, where the role of the force field is played by shape. Thanks to this property, object-based representation brings a very important feature at no cost, called *interactivity*. Interactivity is defined by some as the element that defines multimedia [5]. This is one of the main reasons for which an object-based representation was adopted in the MPEG-4 standard; see Chapter 6.5 and [6]. As pointed out earlier, because of the fact that the majority of digital visual information is still in pixel-based representations, converters are needed in order to go from one representation to another. The passage from a pixel-based representation to an object-based representation can be performed by using manual, semiautomatic, or automatic segmentation techniques. This subject will not be covered here, as it is addressed in Chapter 4.8. The inverse operation is achieved by rendering, blending, or composition, which are typically used in computer graphics applications. Object-based representations are also very suitable to be cast in the same framework as natural and synthetic data coding, since synthetic objects (2-D or

3-D) can be treated in the same way as any natural object and added into the scene (see Fig. 1). A large number of object-based coding schemes have been proposed in the literature. The main differences among these techniques reside in one of the following points:

- the specific method used for the **coding of** shapes of objects
- the method used for the coding **of texture** and color information in objects
- the method used to estimate and to code the motion of objects
- the way in which the complete system is integrated, using the above components

In this chapter, we will not cover all possible variants and approaches to object-based video coding. Interested readers can refer to tutorial articles and books for this purpose [18, 20]. Rather, the remainder of this chapter will concentrate on object-based video coding algorithms that provide major functionalities expected from such an approach while providing other useful features.

In the data representation pyramid, one could think of yet another representation in which visual information is represented by describing its content. An example would be when you describe to someone a person he or she has never seen: She is tall, thin, has long black hair, blue eyes, etc. As this kind of representation would require some degree of semantic understanding, one could call it a "semantics-based representation." One way of building a semantics-based representation is to start from an object-based or content-based representation, as again, it seems that humans do it this way [1, 2]. An example of an implementation of a semantics-based representation would be a descriptor language that describes objects and their properties (position, dominant color, texture, shape, etc.), as well as their relations to each other (close, far, connected, above, etc.). The semantic description can be based on other simpler semantic descriptors in a hierarchical manner. For instance, a house could be by itself a semantic descriptor, which can be also divided into other semantic descriptors such as doors, windows, roof, walls, etc., which could each be divided into simpler semantic descriptors (geometric objects with various shapes, colors and textures, etc.). The difficulty in a semantics-based representation is to make the description as application independent as possible. The coding scheme described in this chapter provides a mechanism that allows efficient access to the salient visual information in an image sequence that is useful for semantics-based representation, while still providing other features desired in a content-based and object-based representation, such as interactivity with objects and compression efficiency.

3 Object-Based Video Coding

This section describes a complete object-based video coding scheme that addresses many requirements desired in applications that would necessitate a content-, object-, or even semantics-based representation. It starts by giving a general overview of the algorithm used for the coding of arbitrarily shaped video objects. The general block diagram of this technique is depicted in Fig. 2. As in other object-based coding schemes, one would expect to distinguish three key components, namely, shape, motion, and texture coding blocks. In this scheme, shape coding is replaced by geometric coding, which refers to information about the outline of objects (shape) as well as its internal visual primitives (edges, corners, etc.).

In addition to the above, as in many video coding schemes, the algorithm operates either in intramode (I), when an object

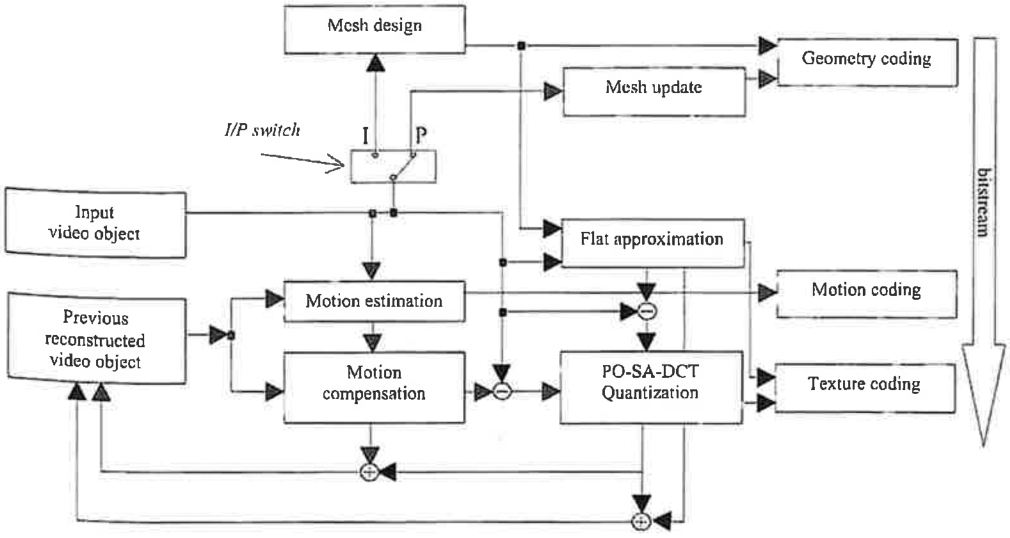


FIGURE 2 Overview of the object-based video object coding structure.

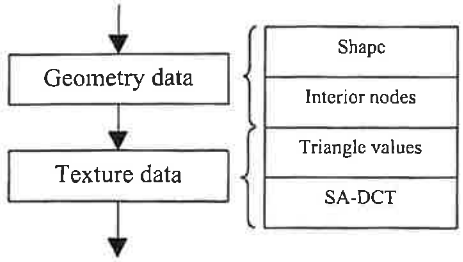


FIGURE 3 Overview of the intracoding mode syntax.

is coded independently, or in a predicted (P) intermode, when a video object is coded taking into account information available on its past. Intramode provides random access points in the bit stream, as well as some robustness to propagation of transmission errors as it does not refer to any prior information. Ideally, intracoded video objects should occur at each scene change, when new objects appear. In practice, they occur at a predefined, fixed rate, e.g., every 0.5 s. Figure 3 gives an overview of the intracoding mode syntax. The geometric information is coded first. The object shape is encoded by using a progressive polygonal approximation, which amounts to a simple vertex coding method when a lossy representation or quasi-lossless shape coding is good enough. Given the mesh outer boundary, interior nodes are selected at high-gradient points by using a minimum distance constraint. The object's outer boundary and inner vertices form a triangular mesh, which is described by coding each vertex position. The entropy associated with a vertex position is in general a function of the size of the video object. By taking into account the forbidden positions, one can reduce this entropy and consequently the amount of bits needed to code the geometry.

Once the geometry (mesh) is coded, the mean color value of each triangle is directly transmitted. The pointwise difference between the original image and the mean value constitutes the texture error image. A shape-adaptive DCT is applied to encode the resulting zero-mean triangular error patches. The transform is followed by uniform quantization of the AC coefficients, zigzag scan, run-length representation, and adaptive arithmetic coding, as in MPEG. At the decoder side, the inverse operations are applied.

Figure 4 gives an overview of the intermode coding syntax. First, the geometric update is encoded: a list of deleted boundary vertices, a list of inserted boundary vertices and their positions, shape motion vectors for predicted vertices, and texture motion vectors for every node (boundary as well as interior). The sign and the absolute value of each vector components are encoded separately with an adaptive arithmetic code (Chapter 5.1), and a special value is defined to indicate that the node is deleted. Then, texture updates are encoded. For each triangle, a one-bit flag indicates to the decoder whether it is updated or not. The choice of whether to update a triangle or not depends on its error measure and a threshold value that is a function of targeted bitrate or desired quality. To perform an update, the same shape-adaptive DCT is applied to each error triangle, combined with uniform quantization of the AC coefficients, zigzag scan, run-length representation, and adaptive arithmetic coding, as in the intramode. At the decoder side, motion compensation and the inverse DCT are applied.

It is important to mention at this point that in addition to a mechanism to generate video objects (by manual, supervised, semiautomatic, or fully automatic segmentation), the encoder should also design a content-based mesh on the video objects by selecting nodes on high spatial gradient points such as those described in [7, 9, 13]. In this case, an adaptive triangular mesh partition is constructed from the resulting set of nodes by means of Delaunay triangulation [17]. Only the node positions (mesh geometry) need to be transmitted for the decoder to reconstruct this content-based partition. If an arbitrarily shaped video object is considered, its outer boundary is approximated by a polygon (vertices), transmitted to the decoder, and constrained Delaunay triangulation is applied. Consecutive occurrences of video objects are predicted by means of forward node tracking. Motion compensation is based on an affine triangular warping model where the motion of any pixel is linearly interpolated from that of surrounding triangle vertices. Only the node motion vectors need to be determined and transmitted to the decoder to track the mesh deformation along the video sequence.

The bitstream syntax is organized in a separable fashion, so as to allow efficient and independent access to geometry (and shape), motion, and texture information in a quality-scalable way, so that the salient information comes first and can be decoded without the need to reconstruct all of the data. Salient information includes a coarse shape description by polygon vertices; mesh node positions (which are selected based on specific image features, such as edges and corners); coarse texture data in intraframes (for instance, one DC component per mesh triangle — flat image approximation); and coarse mesh motion (defined by the tracking trajectories of a limited set of significant vertices). This codec provides many functionalities needed for video compression, video object coding, and manipulation, as well as content-based retrieval in video databases [13].

In the following paragraphs the major components of this coding algorithm are described in further detail, and more insights are provided.

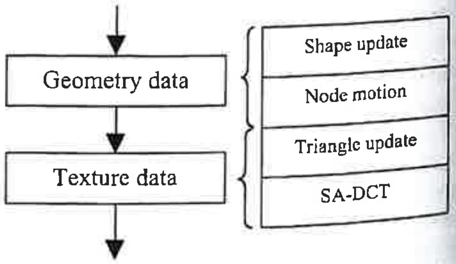


FIGURE 4 Overview of the intercoding mode syntax.

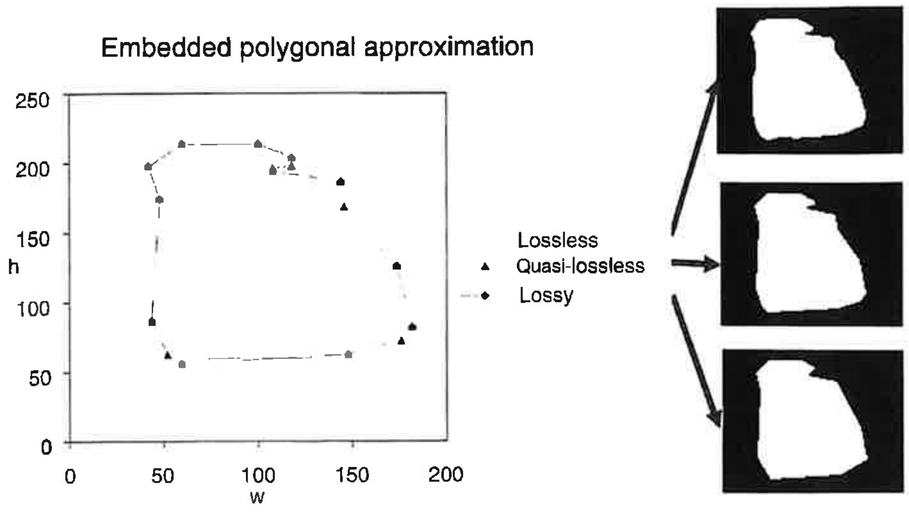


FIGURE 5 Example of video object decoding, using PPE from coarse to fine to lossless. (See color section, p. C–28.)

3.1 Object Shape and Geometry Coding

In object-based coding techniques, information about the shape of video objects has to be coded and made available in the bitstream. This component is the major difference between object-based and more conventional pixel-based techniques, as the shape information is not needed in the latter and is therefore not coded. A progressive contour coding based on a polygonal approximation of the shape boundary is used to code the outline of every video object [12]. The corresponding progressive polygonal encoding (PPE) method exploits the previously transmitted coarser polygons to achieve efficient compression of subsequent contour refinements, defined either geometrically or by a chain code (when lossless shape coding is desired). This representation offers several interesting features. First, being quality scalable, geometrical, and semantic, it is particularly suitable for sketch-based retrieval that is based on video object shapes, as well as for video manipulation. Indeed, the decoder can easily decode the first bits in the shape bit stream that correspond to the most salient vertices, typically high-curvature points along the shape contour. Figure 5 gives an example of a video object that has been decoded in a progressive manner from coarse to fine, and up to a lossless level. Shape matching methods such as vertex-based modal matching or comparisons based on the Hausdorff distance can then directly exploit this coarse vertex representation [10, 13]. Second, a geometrical shape boundary description can be integrated into an object-based mesh coding scheme. Coarse vertices simply define the outer mesh boundary, and constrained Delaunay triangulation can be applied to define a corresponding arbitrarily shaped triangular mesh partition [1, 9, 17, 19].

In order to support lossless shape representation, as required by high-quality applications for appropriate object texture rendering, a solution has to be designed to efficiently and losslessly compress video object shapes while maintaining a reasonable complexity. Such a solution is based on *altered* boundary triangles, which is enabled by a specific property of the PPE representation: the lossless contour refinement is constrained into a geometrical stripe one or two pixels wide on both sides of the coarser polygonal approximation, if the latter was defined under an accuracy of one or two pixels respectively. The boundary triangles in the mesh can then be easily adapted to fit the lossless shape boundary, by checking only pixels in a thin stripe along the mesh boundary, as illustrated by Fig. 6. A detailed example of

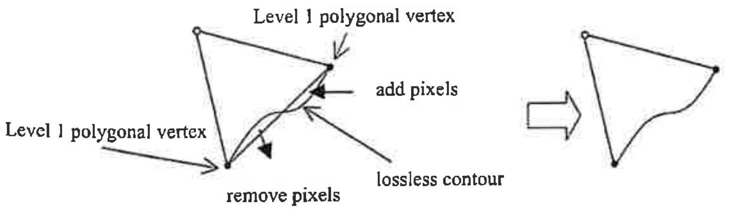


FIGURE 6 Lossless shape refinement with altered triangles. Left: mesh triangle and its corresponding original contour, which is no farther than one pixel away from the boundary edge; right: it is possible to obtain the original shape by adding and removing pixels where necessary.

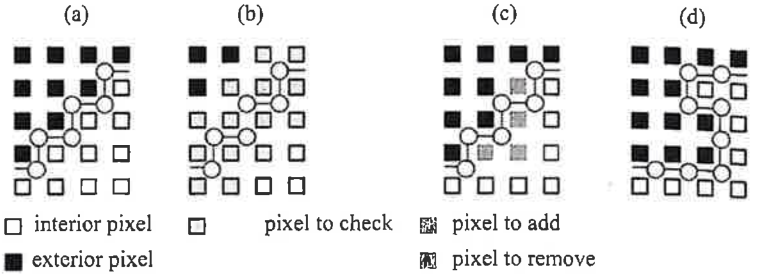


FIGURE 7 Example of refinement along a boundary edge, resulting in an altered triangle. Squares indicate 2-D pixels; circles and lines represent the interpixel discrete segment. (a) Coarse shape boundary; (b) local stripe to refine; (c) refined pixels; (d) final lossless shape boundary.

this stripe-based boundary refinement is given in Fig. 7. Figure 8 shows the coarse mesh, the refined mesh, and the corresponding updated pixels for a mesh-based partition of a typical video object.

In the intermode, a temporally predicted shape can be used in order to reduce the shape coding overhead and to take advantage of temporal correlation regarding the contour information. To this end, the progressive polygonal approximation method is applied to each occurrence of video objects, and the resulting coarse vertices are matched to the previous corresponding video objects (polygon matching). Information about deleted, inserted, and tracked vertices is sent to the decoder in the form of binary lists, followed by the prediction motion vector or the inserted vertex position depending on the transmitted vertex status. Refinement vertices are still intra-encoded by means of the PPE algorithm, as they are likely to correspond to details that are expected to be temporally unstable. Experimental results show a gain of about 40% by using such predicted shape coding schemes when compared with intrashape coding for rigid and even slightly nonrigid video objects [13].

3.2 Object Motion Estimation, Compensation, and Coding

In triangular mesh-based video codecs, the motion at each node defining the mesh is determined and transmitted to the decoder, which applies affine warping as the motion compensation method to interpolate the motion in each mesh triangle [8, 9, 15, 16, 19]. Various node motion estimation methods have been investigated so far in the literature. The simplest technique consists of performing block matching by defining a square block centered on the node to track. Forward or backward block matching may be used [15], the former being more suited to node trajectory tracking along the video sequence [19]. A variant of this method, called pixel matching, consists of weighting the error computation in the block matching process so that higher importance is given to the error at and immediately around the node itself, as the aim is a node motion estimation rather than a block motion estimation [15, 16]. Experimental results show that block matching methods outperform pixel matching algorithms in terms of motion compensation quality, especially in the presence of mild to complex motion [13]. The major drawback of block matching as well as pixel matching lies in the fact that they do not take into account the affine warping process in the motion optimization. Consequently, the compensation error is not exactly the computed error in the motion estimation procedure, and a suboptimal solution may be obtained.

To overcome this limitation, two major methods have been reported in the literature: closed-form connectivity-preserving solutions [8] and hexagonal matching refinement [14]. The first method operates on a dense optical flow field, possibly derived from a prior video segmentation and tracking stage. The dense motion field requirement together with its relative complexity explain its infrequent use in practice. The hexagonal matching refinement method aims at taking into account the warping-based motion compensation in the motion estimation process. It was initially applied to regular (hexagonal) triangular partitions [14], but several authors have adapted it to content-based

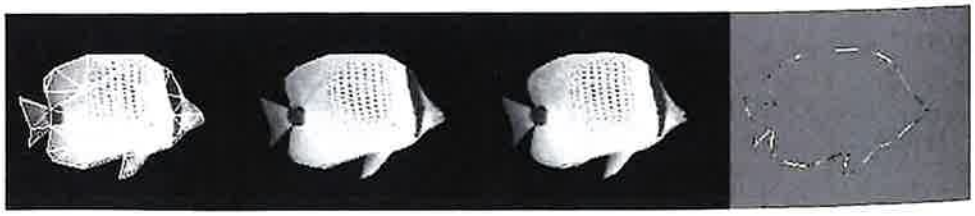


FIGURE 8 Left: triangular mesh partition; center left: coarse boundary; center right: exact boundary with altered triangles; right: pixels processed in a shape-refinement process (black: removed pixels; white: added pixels).

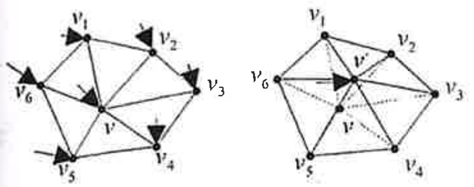


FIGURE 9 Hexagonal matching refinement. Left: triangulation is applied based on the positions of predicted refinement nodes and previously determined coarser nodes; right: the refinement node position is actually optimized to minimize the warping error in the corresponding surrounding polygons.

triangular meshes, fitting arbitrarily shaped video objects [9, 16]. This approach relies on an initial guess for mesh node motion, possibly provided by a block matching technique. Based on this initial solution, the motion vector at each node is optimized by minimizing the affine compensation error in connected triangles, assuming the motion of connected nodes is fixed (see Fig. 9). The optimization is repeated for each node and iterated over the whole set of nodes until stability is reached. Each pass of the algorithm guarantees that the error decreases when compared with the initial guess error. However, in addition to its complexity, the algorithm may also suffer from other limitations, such as its suboptimality and high sensitivity to initial guess values, as outlined in [22]. In practice, while being much more complex in terms of both implementation and computation, the hexagonal matching refinement method does not necessarily generate better results than the direct block matching method. The latter is therefore preferred as the motion estimation algorithm, and it is used here (and for the base layer in the progressive coding scheme described further).

3.3 Texture Representation

In a complete video compression scheme, texture approximation and encoding are needed in the intracoding mode, but also in intercoding modes when prediction residual errors should be coded. Until very recently, mesh-based video representations were often embedded in standard video coding schemes with little attention devoted to their efficient integration [21]. The present object-based coding algorithm is designed to provide a complete and consistent video compression scheme, where the texture representation method is suitable to the triangular partition associated with the warping motion model and applicable to both intracoding and intercoding modes.

In classical intraframe mesh-based image approximation methods, intensity values are transmitted only at the mesh nodes, and other pixel values are interpolated from them; for instance, by means of an affine model applied to the mesh triangles. The major drawback of this approach is the underlying assumption of a continuous image surface, which clearly does not support edge and contour discontinuities. As pointed out earlier, a flat approximation is used to represent the intensity of each triangle which results in coding one value per mesh triangle. Figure 10 shows an example of such a coarse representation, based on nodes selected on features such as edges and corners. This coarse field corresponds to salient features easily accessed in the bit stream and suitable for fast discrimination between images, for instance in a content-based retrieval scheme [13]. The corresponding reconstructed image is then further refined by means of transform coding of the residual texture error.

In order to efficiently approximate and encode the texture data in intracoding as well as intercoding modes, a transform method is used. Such methods are very popular in image and video compression. Their major drawback lies in the fact that they were originally designed for pixel-based compression of rectangular images, as opposed to content-based approaches. However, with the emergence of the MPEG-4 standard, different solutions have been recently proposed that partly overcome this problem, such as padding and shape-adaptive transforms [6, 11].

In the framework of mesh-based compression, both Wang [21] and Altunbasak [7, 8] have reported the use of quadrilateral warping combined with conventional block-based DCT. However, the major drawback of this method lies in the additional low-pass filtering effect introduced in the compression scheme by the direct and inverse digital warping procedures [22]. Therefore, rather than transforming the triangles to fit a quadrilateral region over which conventional transforms may be applied, another approach consists of directly applying a transform to the triangular domain, such as the pseudo-orthonormal shape-adaptive discrete cosine transform (PO-SADCT) [11]. With such a transform, there are as many coefficients to code as there were pixels in the shape. In addition, these coefficients are gathered in the top-left part of the shape's bounding box, which makes further quantization and run-length coding similar to the conventional DCT coding scheme. The decoder only needs the shape information to apply the inverse operations and reconstruct the approximated segment. The efficiency of the SADCT has been assessed in the case of 8 × 8 boundary blocks (conventional blocks partly overlapping the border of a video object), for both intra-texture and displaced frame difference (DFD) coding. Variants of the SADCT method have been described in [11]. Among them,

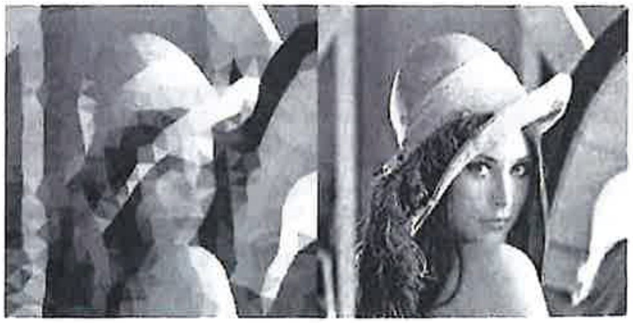


FIGURE 10 Example of application of texture coding. Left: flat approximation (mean or DC intensity values of mesh triangles); Right: PO-SADCT coding of remaining AC coefficients per triangle (compression ratio, 32:1; PSNR, 30.6 dB).

the PO-SADCT method has been shown to be suitable for coding zero-mean texture data, such as DFD and error coding in general. Here, this transform is applied to partition triangles corresponding to intracoding mode as well as intercoding mode prediction residual errors. It is then followed by conventional uniform quantization, zigzag scan, run-length representation, and adaptive arithmetic coding, as in MPEG.

Figure 10 shows an example of the application of this texture coding scheme and its coding efficiency for compression of a still picture.

4 Progressive Object-Based Video Coding

The object-based video compression scheme discussed in the previous section may also be adapted to achieve progressive coding desired in a number of applications. As contour and texture coding techniques used in this coding algorithm are both inherently progressive, the key to achieve an overall progressive coding scheme would be to adopt a progressive geometry (mesh) and motion coding.

Let us consider a progressive mesh from its coarsest to finest levels. At the coarsest level, the mesh is designed as usual, using a minimum distance constraint. By progressively reducing this constraint, one may define new nodes along image edges. This technique enables a progressive mesh design. The shape accuracy may be improved at the same time by using the PPE method. Examples of a few levels of a progressive mesh built on top of a typical video object according to the above process are given in Fig. 11. When encoding the location of a node at a given resolution level, some positions within an 8-neighborhood around a previously transmitted node (contour or interior node, from coarser levels or from the current layer) are invalid. The entropy associated with this node location is therefore reduced accordingly, as well as the number of bits needed for its coding.

As the mesh node density progressively increases, the quality of the resulting motion approximation will improve, as long as the mesh has not reached the optimal size [9, 13]. The motion coding cost also increases with the number of motion vectors to transmit. It may therefore be interesting to first transmit the major node trajectories, then progressively refine them as more bits become available. In order to improve the rate-distortion performance, it is possible to exploit the previously transmitted motion information when encoding the refinement motion vectors. In particular, under the hypothesis of a reasonably smooth motion field, coarse motion vectors can be used as predictors of the refinement trajectories, as illustrated by Fig. 12. In terms of compression, such a prediction will be efficient as long as the hypothesis of a smooth motion field remains true. Indeed, in this case, if motion estimation and resulting motion vector coding are performed relative to the predicted position, rather than the reference position, null and small displacements become more likely.

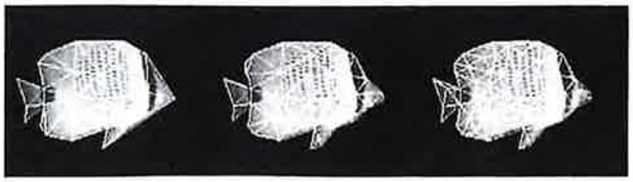


FIGURE 11 Example of a progressive geometry (mesh) construction by successive refinements from left to right.

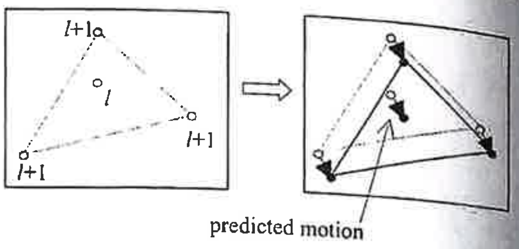


FIGURE 12 Example of progressive motion prediction. Once the motion vectors for level $l+1$ nodes are known, it is possible to predict motion vectors for level l nodes.

By enabling a suitable motion prediction for refinement data at both the encoder and decoder sides, progressive motion transmission also facilitates local optimization of the refinement motion vectors. Indeed, under the hypothesis that the initial prediction derived from coarser motion vectors provides a satisfactory initial guess, triangulation can be applied at this stage. Refinement motion vectors can then be further optimized by means of hexagonal matching refinement, as illustrated in Fig. 9. As opposed to direct block matching, this method takes into account the warping-based rendering process in the optimal displacement computation. As explained earlier, its major drawbacks lie in its inherent complexity and in the fact that it imposes *a priori* triangulation, which is suboptimal when the initial guess is far from the local optimum. By predicting the refinement motion from displacement vectors at surrounding coarser nodes, however, the initial guess is expected to be close enough to the final solution. In addition, many nodes corresponding to coarser motion and contour approximation are fixed, which reduces the search space and accordingly the necessary computation. In particular, if all the nodes connected to a refinement node are fixed (e.g., nodes v_1 to v_6 in Fig. 9), there is no need to iterate the optimization on the corresponding polygon, which accelerates convergence. For the reasons mentioned above, hexagonal matching is performed for estimation of motion vectors of a finer resolution mesh from a coarser one. Experiments show that this approach produces superior results in terms of rate distortion when compared with other motion estimation methods [13].

A complete progressive scheme can then be designed by combining progressive shape, geometric, and motion coding with texture refinement, applied at each layer to the error image generated at the previous coarser layer and cropped by the refined shape mask [13]. Such a progressive transmission scheme may be required when both high-quality decoders and low-cost decoders receive the encoded bit stream, so that the former can take advantage of all of the data while the latter only use the

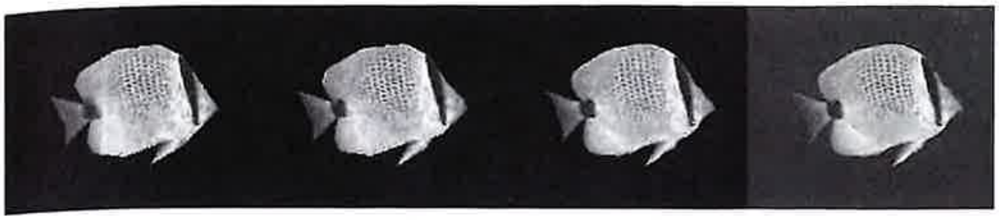


FIGURE 13 Example of progressive decoding of a typical video object. From left to right are decoded results from lowest quality (70 kbps) to medium quality (170 kbps), to highest quality (320 kbps). The original video object is shown to the right side.

oarser part of it. Independent of the decoder performance, scalble transmission also provides decoders with the possibility to uickly browse and preview a coarse version of the video, at a action of the resources required by complete decoding.

Figure 13 depicts results from a progressive decoding of a ypical video object using the described coding algorithm. When he progressive representation is achieved by means of a qualityalable scheme carefully designed to this end, the availability of igh-level, preorganized, and easy-to-access information at the oarse data layer enables specific processing at the decoder side. particular, it facilitates sequence matching, indexing, retrieval, lassification, and automatic event control. In such a scenario, art of the analysis stage performed at the encoder for the sake f compression can be exploited (and saved) at the decoder side, uch as contour extraction or motion analysis. In this coding gorithm, examples of information potentially exploitable at is level include the following: the shape contour information, ither accurate or approximated by a set of vertices; the mesh eometry, where nodes are defined along edges and in highly xtured areas; the motion vectors; the node trajectories; and the oarse color representation.

5 Dynamic Coding

is well known that visual information has a highly nonstaonary nature. In multimedia applications, all sort of visual ta could be transmitted between terminals. Among all the chniques already investigated in the literature, some perform tter in particular regions of an image than others. Typically, bband/wavelet schemes are known to perform well in areas ith texture, whereas techniques based on object representation, morphological operators perform well in areas with sharp ges and contours. Similarly, methods using linear transforms oduce poor results in areas with text or graphics. Dynamic ding is a solution to solve the drawbacks existing in a given heme while still maintaining its strong performance where ppropriate. The basic idea behind dynamic coding is simple t powerful [23]. The visual information (a frame of video, or video object) is subdivided into several regions with similar itability for a given compression method. Each region is encoded by using a multitude of compression techniques. Among these techniques, the one which is the most efficient is chosen, and the compressed bit stream of the region using the best coding technique is sent to the decoder along with information specifying which technique was chosen for its coding. As an example, in areas with texture, a subband/wavelet technique would be used, while areas with strong edges and contours will be coded with morphological-based or other more appropriate techniques. Similarly, text areas will use an encoding technique more appropriate for an efficient compression of such data.

The concept of dynamic coding implicitly defines a general coding syntax. Video objects are further segmented into regions, each represented by their respective representation model. The syntax therefore relies on two degrees of freedom, namely, the video object partition into its constituting regions and their associated representation models.

As depicted in Fig. 14, the resulting syntax is both open and flexible. Indeed, different classes of partitioning can be considered, ranging from the whole image as a single object to arbitrarily shaped video objects segmented into regions of predefined or arbitrary shapes. Additionally, each region resulting from a particular segmentation can be coded with respect to a model chosen from a multitude of representation methods. Figure 15 gives an example of dynamic coding of a rectangular still image by putting in competition a linear and a nonlinear subband decomposition scheme. As it can be seen from this figure, the highly texture regions are best represented when a linear filter bank is used for subband decomposition, while sharp edges and contours are better maintained by using a nonlinear filter bank. A dynamic approach applied to this image allows the use of the best configuration in the region where it is appropriate and produces the best results.

6 Conclusions

In this chapter an object-based video coding scheme was presented that supports arbitrarily shaped video objects, possibly with a lossless shape accuracy. To this end, a progressive polygonal contour approximation is integrated in a complete, consistent, coding scheme. In this context, various node motion estimation methods are used, and the application of the shape-adaptive DCT transform to residual error representation in a content-based, triangular mesh partition is described. The adaptation of this scheme to achieve progressive compression was

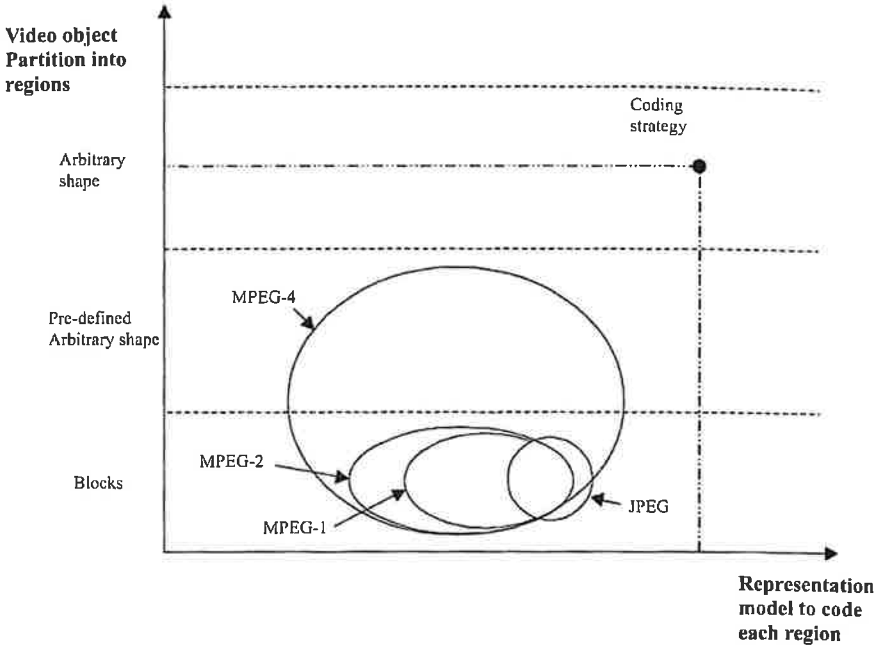

Video object Partition into regions
Coding strategy
Arbitrary shape
Pre-defined Arbitrary shape
Blocks
MPEG-4
MPEG-2
MPEG-1
JPEG
Representation model to code each region

FIGURE 14 Dynamic coding principle.

also discussed and solutions were presented for geometric and motion representations. As opposed to standard compression schemes, including MPEG-4, the proposed scheme supports a separate, content based, quality-scalable syntax where the most salient information is transmitted first and the shape, motion, and texture information fields can be accessed separately in the bit stream. This hierarchical, semantic organization of the encoded data is of particular interest for content-based indexing and retrieval applications, while still allowing an efficient implementation of this scheme for compression and object-based interactivity.

The second part of the chapter briefly introduced the notion of dynamic coding of visual information. Dynamic coding offers the opportunity of combining several compression techniques on different objects or regions where most appropriate. It was shown in a simple example how a dynamic coding approach can provide superior results by a clever combination of algorithms in regions where specific compression techniques produce better results.

Acknowledgment

The object-based video coding scheme presented in this chapter is the result of the Ph.D. thesis of Corinne Le Buhan. Many materials of this chapter benefited from her work. The dynamic

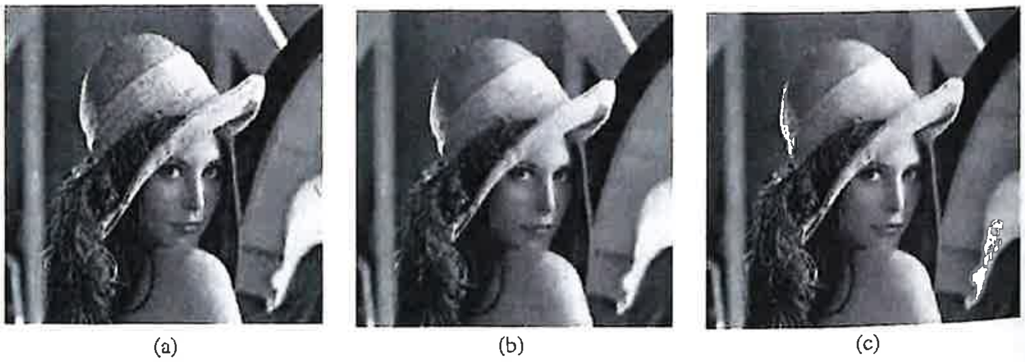


FIGURE 15 Dynamic coding of a still picture at a compression factor of 50:1. (a) Compressed by a linear subband decomposition; (b) compressed by a nonlinear subband decomposition; (c) compressed by dynamic coding that pits the two methods against each other.

coding results were obtained in the framework of the Ph.D. theses of Olivier Egger and Emmanuel Reusens. Both authors acknowledge the valuable inputs from these sources.

References

[1] D. Marr, *Vision — A Computational Investigation into the Human Representation and Processing of Visual Information* (Freeman, San Francisco, 1982, ISBN 0-7167-1567-8).

[2] L. Spillmann and J. S. Werner, *Visual Perception — The Neurophysiological Foundations* (Academic, New York, 1990, ISBN 0-12-657676-9).

[3] M. Kunt, A. Ikonomopoulos, and M. Kocher, "Second generation image coding techniques," *Proc. IEEE* **73**, 549–675 (1985).

[4] R. Castagno, "Video segmentation based on multiple features for interactive and automatic multimedia applications," Ph.D. thesis 1894 (EPFL, Lausanne, Switzerland, 1998).

[5] N. Negroponte, *Being Digital* (Hodder & Stoughton, London, 1995).

[6] T. Ebrahimi, "MPEG-4 Video Verification Model: a video encoding/decoding algorithm based on content representation," *Signal Process. Image Commun.*, Special issue on MPEG-4, **9**, 367–384 (1997).

[7] Y. Altunbasak and A. M. Tekalp, "Scalable mesh-based interpolative coding of synthetic and natural image objects," in *Visual Communications and Image Processing, Proc. SPIE* **3024**, 1004–1011 (1997).

[8] Y. Altunbasak and A. M. Tekalp, "Closed-form connectivity preserving solutions for motion compensation using 2-D meshes," *IEEE Trans. Image Process.* **6**, 1255–1269 (1997).

[9] M. Dudon, O. Avaro, and C. Roux, "Triangular active mesh for motion estimation," *Signal Process. Image Commun.* **10**, 21–41 (1997).

[10] B. Gunsel, A. M. Tekalp, and P. J. L. van Beek, "Moving visual representations of video objects for content-based search and browsing," *IEEE Int. Conf. Image Proc.* **2**, 502–505 (1997).

[11] P. Kauff and K. Schuur, "Shape-adaptive DCT with block-based DC separation and ΔDC correction," *IEEE Trans. Circuits Syst. Video Technol.* **8**, 237–242 (1998).

[12] C. Le Buhan Jordan, T. Ebrahimi, and M. Kunt, "Progressive content-based shape compression for retrieval of binary images," *Computer Vis. Image Understand.*, Special issue on computer vision applications for network-centric computing, **71**, 198–212 (1998).

[13] C. Le Buhan, "Progressive geometrical compression of arbitrary shaped video objects," Ph.D. thesis 1891 (EFPL, Lausanne, Switzerland, 1998; http://ltswww.epfl.ch).

[14] Y. Nakaya and H. Hirashima, "Motion compensation based on spatial transformations," *IEEE Trans. Circuits Syst. Video Technol.* **4**, 339–356 (1994).

[15] J. Nieweglowski, T. G. Campbell, and P. Haavisto, "A novel video coding scheme based on temporal prediction using digital image warping," *IEEE Trans. Consumer Electron.* **39**, 141–150 (1993).

[16] K. Schröder, "Vertex tracking for grid-based motion compensation," in *Visual Communications and Image Processing, Proc. SPIE* **2952**, 264–275 (1996).

[17] J. R. Shewchuk, "Triangle: engineering a 2-D quality mesh generator and Delaunay triangulator," Tech. Rep., (Carnegie Mellon University, Pittsburgh, PA, 1996; http://www.cs.cmu.edu/~quake/triangle.html).

[18] L. Torres and M. Kunt, *Video Coding: The Second Generation Approach* (Kluwer, Boston, 1996).

[19] P. J. L. van Beek and A. M. Tekalp, "Object-based video coding using forward tracking 2-D mesh layers," in *SPIE Visual Communications and Image Processing, Proc. SPIE* **3024**, 699–710 (1997).

[20] T. Ebrahimi and M. Kunt, "Visual data compression for multimedia applications," *Proc. IEEE* **86**, 1109–1125 (1998).

[21] Y. Wang, O. Lee, and A. Vetro, "Use of two-dimensional deformable mesh structures for video coding, Part II – the analysis problem and a region-based coder employing an active mesh representation," *IEEE Trans. Circuits Syst. Video Technol.* **6**, 647–659 (1996).

[22] C. Le Buhan, T. Ebrahimi, and M. Kunt, "Progressive mesh-based coding of arbitrary-shaped video objects," in *Visual Communications and Image Processing, Proc. SPIE* **3653**, 1190–1201 (1999).

[23] E. Reusens, T. Ebrahimi, and M. Kunt, "Dynamic coding of visual information," *IEEE Trans. Circuits Syst. Video Technol.* **7**, 489–500 (1997).

6.4

MPEG-1 and MPEG-2 Video Standards

Supavadee Aramvith
and Ming-Ting Sun
University of Washington

1 MPEG-1 Video Coding Standard

1.1 Introduction

1.1.1 Background and Structure of MPEG-1 Standards Activities

The development of digital video technology in the 1980s has made it possible to use digital video compression in various kinds of applications. The effort to develop standards for coded representation of moving pictures, audio, and their combination is carried out in the Moving Picture Experts Group (MPEG). MPEG is a group formed under the auspices of the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC). It operates in the framework of the Joint ISO/IEC Technical Committee 1 (JTC 1) on Information Technology, which was formerly Working Group 11 (WG11) of Sub-Committee 29 (SC29). The premise is to set the standard for coding moving pictures and the associated audio for digital storage media at ~1.5 Mbit/s so that a movie can be compressed and stored in a CD-ROM (Compact Disc-Read Only Memory). The resultant standard is the international standard for moving picture compression, ISO/IEC 11172 or MPEG-1 (Moving Picture Experts Group-Phase 1). MPEG-1 standards consist of five parts, including systems (11172-1), video (11172-2), audio (11172-3), conformance testing (11172-4), and software simulation (11172-5). In this chapter, we will focus only on the video part.

The activity of the MPEG committee started in 1988 based on the work of ISO JPEG (Joint Photographic Experts Group) [1] and CCITT Recommendation H.261: "Video Codec for Audiovisual Services at $p \times 64$ kbits/s" [2]. Thus, the MPEG-1 standard has much in common with the JPEG and H.261 standards. The MPEG development methodology was similar to that of H.261 and was divided into three phases: requirements, competition, and convergence [3]. The purpose of the requirements phase is to precisely set the focus of the effort and determine the rule for the competition phase. The document of this phase is a "proposal package description" [4] and a test methodology [5]. The next step is the competition phase, in which the goal is to obtain state of the art technology from the best of academic and industrial research. The criteria are based on the technical merits and the tradeoff between video quality and the cost of implementation of the ideas and the subjective test [5]. After the competition phase, various ideas and techniques are integrated into one solution in the convergence phase. The solution results in a document called the simulation model. The simulation model implements, in some sort of programming language, the operation of a reference encoder and a decoder. The simulation model is used to carry out simulations to optimize the performance of the coding scheme [6]. A series of fully documented experiments called core experiments are then carried out. The MPEG committee reached the Committee Draft (CD) status in September 1990 and the Committee Draft (CD 11172) was approved in December 1991. International Standard (IS) 11172 for the first three

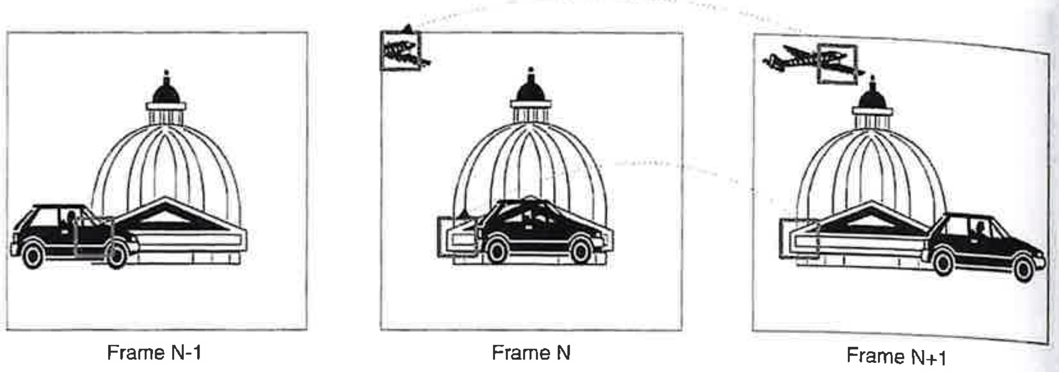


FIGURE 1 A video sequence, showing the benefits of bidirectional prediction.

parts was established in November 1992. The IS for the last two parts was finalized in November 1994.

1.1.2 MPEG-1 Target Applications and Requirements

The MPEG standard is a generic standard, which means that it is not limited to a particular application. A variety of digital storage media applications of MPEG-1 have been proposed based on the assumptions that the acceptable video and audio quality can be obtained for a total bandwidth of ~1.5 Mbits/s. Typical storage media for these applications include CD-ROM, DAT (digital audio tape), Winchester-type computer disks, and writable optical disks. The target applications are asymmetric applications in which the compression process is performed once and the decompression process is required often. Examples of the asymmetric applications include video CD, video on demand, and video games. In these asymmetric applications, the encoding delay is not a concern. The encoders are needed only in small quantities, whereas the decoders are needed in large volumes. Thus, the encoder complexity is not a concern, whereas the decoder complexity has to be low in order to result in low-cost decoders.

The requirements for compressed video in digital storage media mandate several important features of the MPEG-1 compression algorithm. The important features include normal playback, frame-based random access and editing of video, reverse playback, fast forward/reverse play, encoding high-resolution still frames, robustness to uncorrectable errors, etc. The applications also require MPEG-1 to support flexible picture sizes and frame rates. Another requirement is that the encoding process can be performed in reasonable speed by using existing hardware technologies and that the decoder can be implemented by using a small number of chips at low cost.

Because MPEG-1 video coding algorithm is based heavily on H.261, in the following sections we will focus only on those that are different from H.261.

1.2 MPEG-1 Video Coding Versus H.261

1.2.1 Bidirectional Motion Compensated Prediction

In H.261, only the previous video frame is used as the reference frame for the motion compensated prediction (forward prediction). MPEG-1 allows the future frame to be used as the reference frame for the motion compensated prediction (backward prediction), which can provide better prediction. For example, as shown in Fig. 1, if there are moving objects, and if only the forward prediction is used, there will be uncovered areas (such as the block behind the car in Frame N) for which we may not be able to find a good matching block from the previous reference picture (Frame $N-1$). In contrast, the backward prediction can properly predict these uncovered areas since they are available in the future reference picture, i.e., frame $N+1$ in this example. As also shown in Fig. 1, if there are objects moving into the picture (the airplane in the figure), then these new objects cannot be predicted from the previous picture but can be predicted from the future picture.

1.2.2 Motion Compensated Prediction with Half-Pixel Accuracy

The motion estimation in H.261 is restricted to only integer-pixel accuracy. However, a moving object often moves to a position that is not on the pixel grid but between the pixels. MPEG-1 allows half-pixel-accuracy motion vectors. By estimating the displacement at a finer resolution, we can expect improved prediction and, thus, better performance than motion estimation with integer-pixel accuracy. As shown in Fig. 2, since there is no pixel value at the half-pixel locations, interpolation is required to produce the pixel values at the half-pixel positions. Bilinear interpolation is used in MPEG-1 for its simplicity. As in H.261, the motion estimation is performed only on luminance blocks.

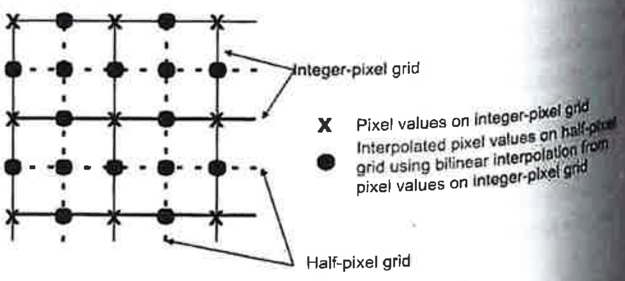

Integer-pixel grid
X Pixel values on integer-pixel grid
● Interpolated pixel values on half-pixel grid using bilinear interpolation from pixel values on integer-pixel grid
Half-pixel grid

FIGURE 2 Half-pixel motion estimation.

The resulting motion vector is scaled by 2 and applied to the chrominance blocks. This reduces the computation but may not necessarily be optimal. Motion vectors are differentially encoded with respect to the motion vector in the preceding adjacent macroblock. The reason is that the motion vectors of adjacent regions are highly correlated, as it is quite common to have relatively uniform motion over areas of picture.

1.3 MPEG-1 Video Structure

1.3.1 Source Input Format

The typical MPEG-1 input format is the source input format (SIF). SIF was derived from CCIR601, a worldwide standard for digital TV studio. CCIR601 specifies the Y Cb Cr color coordinate, where Y is the luminance component (black and white information), and Cb and Cr are two color difference signals (chrominance components). A luminance sampling frequency of 13.5 MHz was adopted. There are several Y Cb Cr sampling formats, such as 4:4:4, 4:2:2, 4:1:1, and 4:2:0. In 4:4:4, the sampling rates for Y, Cb, and Cr are the same. In 4:2:2, the sampling rates of Cb and Cr are half of that of Y. In 4:1:1 and 4:2:0, the sampling rates of Cb and Cr are one quarter of that of Y. The positions of Y Cb Cr samples for 4:4:4, 4:2:2, 4:1:1, and 4:2:0 are shown in Fig. 3.

Converting analog TV signals to digital video with the 13.5-MHz sampling rate of CCIR601 results in 720 active pixels per line (576 active lines for PAL (Phase Alternating Line) and 480 active lines for NTSC (National Television System Committee)). This results in a 720 × 480 resolution for NTSC and a 720 × 576 resolution for PAL. With 4:2:2, the uncompressed bit rate for transmitting CCIR601 at 30 frames/s is then ~166 Mbits/s. Since it is difficult to compress a CCIR601 video to 1.5 Mb/s with good video quality, in MPEG-1, typically the source video resolution is decimated to a quarter of the CCIR601 resolution by filtering and subsampling. The resultant format is called source input format, which has a 360 × 240 resolution for NTSC and a 360 × 288 resolution for PAL. Since in the video coding algorithm the block size of 16 × 16 is used for motion compensated prediction, the number of pixels in both the horizontal and the vertical dimensions should be multiples of 16. Thus, the four leftmost and rightmost pixels are discarded to give a 352 × 240 resolution for NTSC systems (30 frames/s) and a 352 × 288 resolution for PAL systems (25 frames/s). The chrominance signals have half of the above resolutions in both the horizontal and vertical dimensions (4:2:0, 176 × 120 for NTSC and 176 × 144 for PAL). The uncompressed bit rate for SIF (NTSC) at 30 frames/s is ~30.4 Mbits/s.

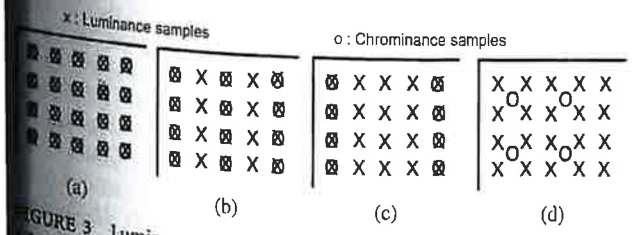


FIGURE 3 Luminance and chrominance samples in formats for (a) 4:4:4, (b) 4:2:2, (c) 4:1:1, (d) 4:2:0.

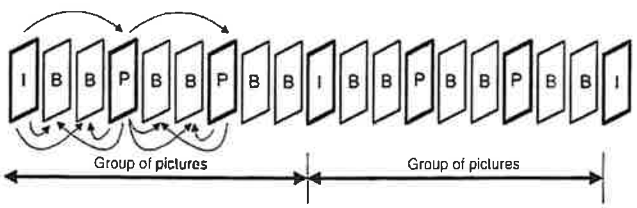


FIGURE 4 MPEG group of pictures.

1.3.2 Group Of Pictures and I-B-P Pictures

In MPEG, each video sequence is divided into one or more groups of pictures (GOPs). There are four types of pictures defined in MPEG-1: I, P, B, and D pictures, of which the first three are shown in Fig. 4. Each GOP is composed of one or more pictures; one of these pictures must be an I picture. Usually, the spacing between two anchor frames (I or P pictures) is referred to as *M*, and the spacing between two successive I pictures is referred to as *N*. In Fig. 4, $M = 3$ and $N = 9$.

I pictures (intracoded pictures) are coded independently with no reference to other pictures. I pictures provide random access points in the compressed video data, since the I pictures can be decoded independently without referencing to other pictures. With I pictures, an MPEG bit stream is more editable. Also, error propagation due to transmission errors in previous pictures will be terminated by an I picture, since the I picture does not have a reference to the previous pictures. Since I pictures use only transform coding without motion compensated predictive coding, it provides only moderate compression.

P pictures (predictive-coded pictures) are coded by using the forward motion-compensated prediction similar to that in H.261 from the preceding I or P picture. P pictures provide more compression than the I pictures by virtue of motion-compensated prediction. They also serve as references for B pictures and future P pictures. Transmission errors in the I pictures and P pictures can propagate to the succeeding pictures, because the I pictures and P pictures are used to predict the succeeding pictures.

B pictures (bidirectional-coded pictures) allow macroblocks to be coded by using bidirectional motion-compensated prediction from both the past and future reference I or P pictures. In the B pictures, each bidirectional motion-compensated macroblock can have two motion vectors: a forward motion vector, which references to a best matching block in the previous I or P pictures, and a backward motion vector, which references to a best matching block in the next I or P pictures as shown in Fig. 5. The motion compensated prediction can be formed by the average of the two referenced motion compensated blocks. By averaging between the past and the future reference blocks, the effect of noise can be decreased. B pictures provide the best compression

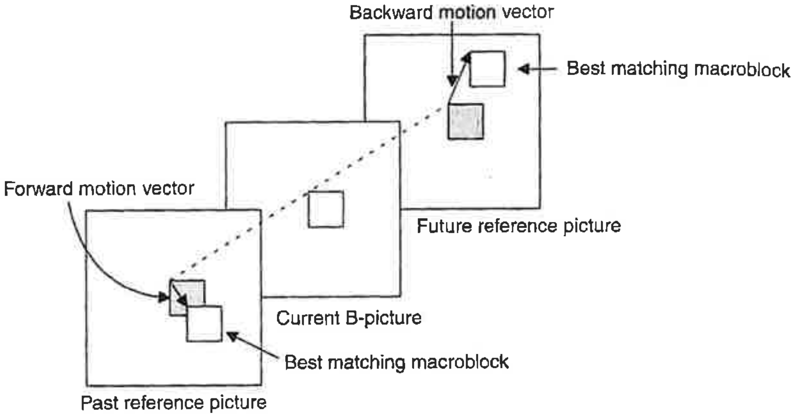


FIGURE 5 Bidirectional motion estimation.

compared to I and P pictures. I and P pictures are used as reference pictures for predicting B pictures. To keep the structure simple and since there is no apparent advantage to use B pictures for predicting other B pictures, the B pictures are not used as reference pictures. Hence, B pictures do not propagate errors.

D pictures (DC pictures) are low-resolution pictures obtained by decoding only the DC coefficient of the discrete cosine transform coefficients of each macroblock. They are not used in combination with I, P, or B pictures. D pictures are rarely used, but they are defined to allow fast searches on sequential digital storage media.

The tradeoff of having frequent B pictures is that it decreases the correlation between the previous I or P picture and the next reference P or I picture. It also causes coding delay and increases the encoder complexity. With the example shown in Fig. 4 and Fig. 6, at the encoder, if the order of the incoming pictures is 1, 2, 3, 4, 5, 6, 7, . . . , the order of coding the pictures at the encoder will be 1, 4, 2, 3, 7, 5, 6, ... At the decoder, the order of the decoded pictures will be 1, 4, 2, 3, 7, 5, 6, ... However, the display order after the decoder should be 1, 2, 3, 4, 5, 6, 7. Thus, frame memories have to be used to put the pictures in the correct order. This picture reordering causes delay. The computation of bidirectional motion vectors and the picture-reordering frame memories increase the encoder complexity.

In Fig. 6, two types of GOPs are shown. GOP1 can be decoded without referencing other GOPs. It is called a closed GOP. In GOP2, to decode the eighth B and ninth B pictures, the seventh P picture in GOP1 is needed. GOP2 is called an open GOP, which means the decoding of this GOP has to reference other GOPs.

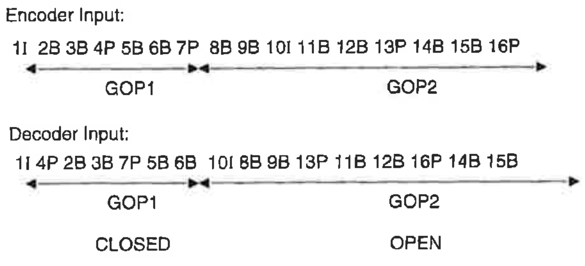

Encoder Input:
1I 2B 3B 4P 5B 6B 7P 8B 9B 10I 11B 12B 13P 14B 15B 16P
GOP1
GOP2
Decoder Input:
1I 4P 2B 3B 7P 5B 6B 10I 8B 9B 13P 11B 12B 16P 14B 15B
GOP1
GOP2
CLOSED
OPEN

FIGURE 6 Frame reordering.

1.3.3 Slice, Macroblock, and Block Structures

An MPEG picture consists of slices. A slice consists of a contiguous sequence of macroblocks in a raster scan order (from left to right and from top to bottom). In an MPEG coded bit stream, each slice starts with a slice header, which is a clear codeword (a clear codeword is a unique bit pattern that can be identified without decoding the variable-length codes in the bit stream). As a result of the clear-codeword slice header, slices are the lowest level of units that can be accessed in an MPEG coded bit stream without decoding the variable-length codes. Slices are important in the handling of channel errors. If a bit stream contains a bit error, the error may cause error propagation because of the variable-length coding. The decoder can regain synchronization at the start of the next slice. Having more slices in a bit stream allows better error termination, but the overhead will increase.

A macroblock consists of a 16 × 16 block of luminance samples and two 8 × 8 block of corresponding chrominance samples as shown in Fig. 7. A macroblock thus consists of four 8 × 8 Y blocks, one 8 × 8 Cb block, and one 8 × 8 Cr block. Each coded macroblock contains motion-compensated prediction information (coded motion vectors and the prediction errors). There are four types of macroblocks: intra, forward predicted, backward predicted, and averaged macroblocks. The motion information consists of one motion vector for forward- and backward-predicted macroblocks and two motion vectors for bidirectionally predicted (or averaged) macroblocks. P pictures can have intra- and forward-predicted macroblocks. B pictures can have all four types of macroblocks. The first and last macroblocks in a slice must always be coded. A macroblock is designated as a skipped macroblock when its motion vector is zero and all the quantized DCT coefficients are zero. Skipped macroblocks are not allowed in I pictures. Nonintracoded macroblocks in P and B pictures can be skipped. For a skipped macroblock, the decoder just copies the macroblock from the previous picture.

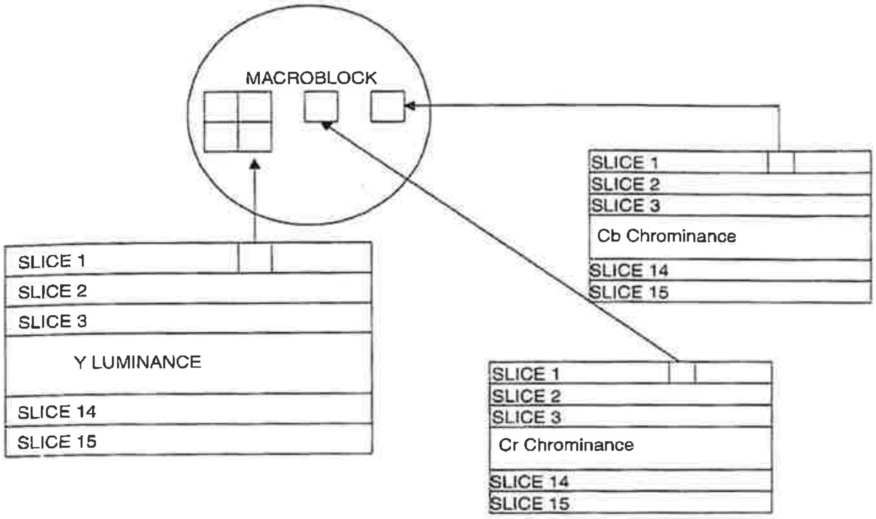


FIGURE 7 Macroblock and slice structures.

1.4 Summary of the Major Differences Between MPEG-1 Video and H.261

Compared with H.261, MPEG-1 video differs in the following aspects:

1. MPEG-1 uses bidirectional motion-compensated predictive coding with half-pixel accuracy, whereas H.261 has no bidirectional prediction (B pictures) and the motion vectors are always in integer-pixel accuracy.
2. MPEG-1 supports the maximum motion vector range of −512 to +511.5 pixels for half-pixel motion vectors and −1024 to +1023 for integer-pixel motion vectors, whereas H.261 has a maximum range of only ±15 pixels.
3. MPEG-1 uses visually weighted quantization based on the fact that the human eye is more sensitive to quantization errors related to low spatial frequencies than to high spatial frequencies. MPEG-1 defines a default 64-element quantization matrix, but it also allows custom matrices appropriate for different applications. H.261 has only one quantizer for the intra-DC coefficient and 31 quantizers for all other coefficients.
4. H.261 only specifies two source formats: CIF (common intermediate format; 352 × 288 pixels) and QCIF (quarter CIF; 176 × 144 pixels). In MPEG-1, the typical source format is SIF (352 × 240 for NTSC, and 352 × 288 for PAL). However, the users can specify other formats. The picture size can be as large as 4k × 4k pixels. There are certain parameters in the bit streams that are left flexible, such as the number of lines per picture (less than 4096), the number of pels per line (less than 4096), picture rate (24, 25, and 30 frames/s), and 14 choices of pel aspect ratios.
5. In MPEG-1, I, P, and B pictures are organized as a flexible group of pictures.
6. MPEG-1 uses a flexible slice structure instead of group of blocks (GOB) as defined in H.261.
7. MPEG-1 has D pictures to allow the fast-search option.
8. In order to allow cost-effective implementation of user terminals, MPEG-1 defines a constrained parameter set, which lays down specific constraints, as listed in Table 1.

1.5 Simulation Model

Similar to H.261, MPEG-1 specifies only the syntax and the decoder. Many detailed coding options such as the rate-control strategy, the quantization decision levels, the motion estimation schemes, and coding modes for each macroblock are not specified. This allows future technology improvement and product differentiation. In order to have a reference MPEG-1 video quality, simulation models were developed in MPEG-1. A simulation model contains a specific reference implementation of the MPEG-1 encoder and decoder, including all the details that are not specified in the standard. The final version of the MPEG-1 simulation model is "simulation model 3" (SM3) [7]. In SM3, the motion estimation technique uses one forward or one backward motion vector per macroblock with half-pixel accuracy. A two-step search scheme, which consists of a full-search in the range of ±7 pixels with the integer-pixel precision, followed

TABLE 1 MPEG-1 constrained parameter set

Parameter	Constraint
Horiz. size	≦720 pels
Vert. size	≦576 pels
Total No. of Macroblocks/picture	≦396
Total No. of Macroblocks/second	≦396 × 25 = 330 × 30
Picture rate	≦30 frames/s
Bit rate	≦1.86 Mbits/s
Decoder Buffer	≦376832 bits

by a search in eight neighboring half-pixel positions, is used. The decision of the coding mode for each macroblock (whether or not it will use motion-compensated prediction and intra or inter coding), the quantizer decision levels, and the rate-control algorithm are all specified.

1.6 MPEG-1 Video Bit-Stream Structures

As shown in Fig. 8, there are six layers in the MPEG-1 video bit stream: the video sequence, group of pictures, picture, slice, macroblock, and block layers.

A video sequence layer consists of a sequence header, one or more groups of pictures, and an end-of-sequence code. It contains the setting of the following parameters: the picture size (horizontal and vertical sizes), pel aspect ratio, picture rate, bit rate, the minimum decoder buffer size (video buffer verifier size), constraint parameters flag (this flag is set only when the picture size, picture rate, decoder buffer size, bit rate, and motion parameters satisfy the constraints bound in Table 1), the control for the loading of sixty-four 8-bit values for intra and nonintra quantization tables, and the user data.

The GOP layer consists of a set of pictures that are in a continuous display order. It contains the setting of the following parameters: the time code, which gives the hours-minutes-seconds time interval from the start of the sequence; the closed GOP flag, which indicates whether the decoding operation requires pictures from the previous GOP for motion compensation; the broken link flag, which indicates whether the previous GOP can be used to decode the current GOP; and the user data.

The picture layer acts as a primary coding unit. It contains the setting of the following parameters: the temporal reference, which is the picture number in the sequence and is used to determine the display order; the picture types (I/P/B/D); the decoder buffer initial occupancy, which gives the number of bits that must be in the compressed video buffer before the idealized decoder model defined by MPEG decodes the picture (it is used to prevent the decoder buffer overflow and underflow); the forward motion vector resolution and range for P and B pictures; the backward motion vector resolution and range for B pictures; and the user data.

The slice layer acts as a resynchronization unit. It contains the slice vertical position where the slice starts, and the quantizer scale that is used in the coding of the current slice.

The macroblock layer acts as a motion compensation unit. It contains the setting of the following parameters: the optional stuffing bits, the macroblock address increment, the macroblock type, quantizer scale, motion vector, and the coded block pattern, which defines the coding patterns of the six blocks in the macroblock.

The block layer is the lowest layer of the video sequence and consists of coded 8 × 8 DCT coefficients. When a macroblock is encoded in the intra mode, the DC coefficient is encoded similar to that in JPEG (the DC coefficient of the current macroblock is predicted from the DC coefficient of the previous macroblock). At the beginning of each slice, predictions for DC coefficients for luminance and chrominance blocks are reset to 1024. The differential DC values are categorized according to their absolute values and the category information is encoded using VLC (variable-length code). The category information indicates the number of additional bits following the VLC to represent the prediction residual. The AC coefficients are encoded similar to that in H.261, using a VLC to represent the zero run length and the value of the nonzero coefficient. When a macroblock is encoded in nonintra modes, both the DC and AC coefficients are encoded similar to that in H.261.

Above the video sequence layer, there is a system layer in which the video sequence is packetized. The video and audio bitstreams are then multiplexed into an integrated data stream. These are defined in the systems part.

1.7 Summary

MPEG-1 is mainly for storage media applications. Because of the use of B picture, it may result in a long end-to-end delay.

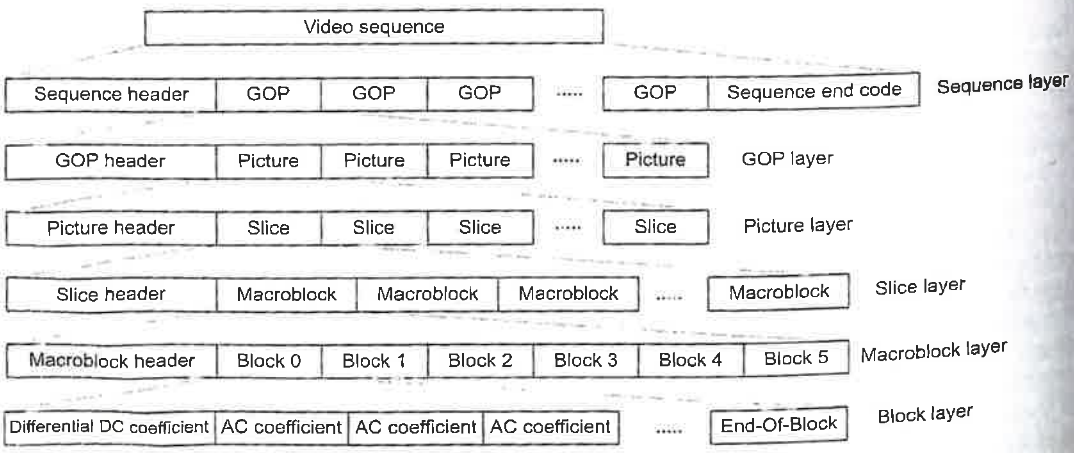


FIGURE 8 MPEG-1 bit-stream syntax layers.

The MPEG-1 encoder is much more expensive than the decoder because of the large search range, the half-pixel accuracy in motion estimation, and the use of the bidirectional motion estimation. The MPEG-1 syntax can support a variety of frame rates and formats for various storage media applications. Similar to other video coding standards, MPEG-1 does not specify every coding option (motion estimation, rate control, coding modes, quantization, preprocessing, postprocessing, etc.). This allows for continuing technology improvement and product differentiation.

2 MPEG-2 Video Coding Standard

2.1 Introduction

2.1.1 Background and Structure of MPEG-2 Standards Activities

The MPEG-2 standard represents the continuing efforts of the MPEG committee to develop generic video and audio coding standards after their development of MPEG-1. The idea of this second phase of MPEG work came from the fact that MPEG-1 is optimized for applications at ~1.5 Mb/s with input source in SIF, which is a relatively low-resolution progressive format. Many higher quality, higher bit-rate applications require a higher resolution digital video source such as CCIR601, which is an interlaced format. New techniques can be developed to code the interlaced video better.

The MPEG-2 committee started working in late 1990 after the completion of the technical work of MPEG-1. The competitive tests of video algorithms were held in November 1991, followed by the collaborative phase. The Committee Draft (CD) for the video part was achieved in November 1993. The MPEG-2 standard (ISO/IEC 13818) [8] currently consists of nine parts. The first five parts are organized in the same fashion as MPEG-1: systems, video, audio, conformance testing, and simulation software technical report. The first three parts of MPEG-2 reached International Standard (IS) status in November 1994. Parts 4 and 5 were approved in March 1996. Part 6 of the MPEG-2 standard specifies a full set of digital storage media control commands (DSM-CC). Part 7 is the specification of a nonbackward compatible audio. Part 8 was originally planned to be the coding of 10-bit video but was discontinued. Part 9 is the specification of real-time interface (RTI) to transport stream decoders, which may be utilized for adaptation to all appropriate networks carrying MPEG-2 transport streams. Parts 6 and 9 have already been approved as International Standards in July 1996. Like the MPEG-1 standard, MPEG-2 video coding standard specifies only bit-stream syntax and the semantics of the decoding process. Many encoding options were left unspecified to encourage continuing technology improvement and product differentiation.

MPEG-3, which was originally intended for HDTV (high-definition digital television) at higher bit rates, was merged with MPEG-2. Hence there is no MPEG-3. The MPEG-2 video coding standard (ISO/IEC 13818-2) was also adopted by ITU-T, as ITU-T Recommendation H.262 [9].

2.1.2 Target Applications and Requirements

MPEG-2 is primarily targeted at coding high-quality video at 4–15 Mb/s for video on demand (VOD), digital broadcast television, and digital storage media such as DVD (digital versatile disc). It is also used for coding HDTV, cable/satellite digital TV, video services over various networks, two-way communications, and other high-quality digital video applications.

The requirements from MPEG-2 applications mandate several important features of the compression algorithm. Regarding picture quality, MPEG-2 has to be able to provide good NTSC quality video at a bit rate of approximately 4–6 Mbits/s and transparent NTSC quality video at a bit rate of approximately 8–10 Mbits/s. It also has to provide the capability of random access and quick channel switching by means of inserting I pictures periodically. The MPEG-2 syntax also has to support trick modes, e.g., fast forward and fast reverse play, as in MPEG-1. Low-delay mode is specified for delay-sensitive visual communications applications. MPEG-2 has scalable coding modes in order to support multiple grades of video quality, video formats, and frame rate for various applications. Error resilience options include intra motion vector, data partitioning, and scalable coding. Compatibility between the existing and the new standard coders is another prominent feature provided by MPEG-2. For example, MPEG-2 decoders should be able to decode MPEG-1 bit streams. If scalable coding is used, the base layer of MPEG-2 signals can be decoded by a MPEG-1 decoder. Finally, it should allow reasonable complexity encoders and low-cost decoders be built with mature technology. Since MPEG-2 video is based heavily on MPEG-1. In the following sections, we will focus only on those features which are different from MPEG-1 video.

2.2 MPEG-2 Profiles and Levels

MPEG-2 standard is designed to cover a wide range of applications. However, features needed for some applications may not be needed for other applications. If we put all the features into one single standard, it may result in an overly expensive system for many applications. It is desirable for an application to implement only the necessary features to lower the cost of the system. To meet this need, MPEG-2 classified the groups of features for important applications into profiles. A profile is defined as a specific subset of the MPEG-2 bit-stream syntax and functionality to support a class of applications (e.g., low-delay video conferencing applications, or storage media applications). Within each profile, levels are defined to support applications that have different quality requirements (e.g., different resolutions). Levels are specified as a set of restrictions on some of the parameters (or their combination) such as sampling rates, frame dimensions, and bit rates in a profile. Applications are implemented in the allowed range of values of a particular profile at a particular level.

TABLE 2 Profiles and levels

	Profile				
Level	Simple 4:2:0	Main 4:2:0	SNR scalable 4:2:0	Spatially scalable 4:2:0	High 4:2:0 or 4:2:2
High 1920 × 1152 (60 frames/s)		62.7 Ms/s 80 Mbit/s			100 Mbit/s for 3 layers
High-1440 1440 × 1152 (60 frames/s)		47 Ms/s 60 Mbit/s		47 Ms/s 60 Mbit/s for 3 layers	80 Mbit/s for 3 layers
Main 720 × 576 (30 frames/s)	10.4 Ms/s 15 Mbit/s	10.4 Ms/s 15 Mbit/s	10.4 Ms/s 15 Mbit/s for 2 layers		20 Mbit/s for 3 layers
Low 352 × 288 (30 frames/s)		3.04 Ms/s 4 Mbit/s	3.04 Ms/s 4 Mbit/s for 2 layers		

Table 2 shows the combination of profiles and levels that are defined in MPEG-2. MPEG-2 defines seven distinct profiles: simple, main, SNR scalable, spatially scalable, high, 4:2:2, and multiview. The last two profiles were developed after the final approval of MPEG-2 video in November 1994. Simple profile is defined for low-delay video conferencing applications. Main profile is the most important and widely used profile for general high-quality digital video applications such as VOD, DVD, Digital TV, and HDTV. SNR (signal-to-noise ratio) scalable profile supports multiple grades of video quality. Spatially scalable profile supports multiple grades of resolutions. High profile supports multiple grades of quality, resolution, and chroma format. Four levels are defined within the profiles: low (for SIF resolution pictures), main (for CCIR601 resolution pictures), high-1440 (for European HDTV resolution pictures), and high (for North American HDTV resolution pictures). The 11 combinations of profiles and levels in Table 2 define the MPEG-2 conformance points that cover most practical MPEG-2 target applications. The numbers in each conformance point indicate the maximum bound of the parameters. The number in the first line indicates the luminance rate in samples/s. The number in the second line indicates bit rate in bits/s. Each conformance point is a subset of the conformance point at the right or above. For example, a main-profile main-level decoder should also decode simple-profile main-level and main-profile low-level bit streams. Among the defined profiles and levels, main-profile at main-level (MP@ML) is used for digital television broadcast in CCIR601 resolution and DVD-video. The main-profile at high-level (MP@HL) is used for HDTV. The 4:2:2 profile is defined to support the pictures with a color resolution of 4:2:2 for higher bit-rate studio applications. Although the high profile supports 4:2:2 also, a high-profile codec has to support SNR scalable profile and spatially scalable profile. This makes the high-profile codec expensive. The 4:2:2 profile does not have to support the scalabilities and thus will be much cheaper to implement. Multiview profile is defined to support the efficient encoding of the application involving two video sequences from two cameras shooting the same scene with a small angle between them.

2.3 MPEG-2 Video Input Resolutions and Formats

Although the main concern of the MPEG-2 committee is to support the CCIR601 resolution, which is the digital TV resolution, MPEG-2 allows a maximum picture size of 16k × 16k pixels. It also supports the frame rates of 23.976, 24, 25, 29.97, 30, 50, 59.94 and 60 Hz, as in MPEG-1. MPEG-2 is suitable for coding progressive video format as well as interlaced video format. As for the color subsampling formats, MPEG-2 supports 4:2:0, 4:2:2, and 4:4:4. MPEG-2 uses the 4:2:0 format as in MPEG-1, except that there is a difference in the positions of the chrominance samples as shown in Figs. 9(a) and 9(b). On one hand, in MPEG-1, a slice can cross macroblock row boundaries. Therefore, a single slice in MPEG-1 can be defined to cover the entire picture. On the other hand, slices in MPEG-2 begin and end in the same horizontal row of macroblocks. There are two types of slice structure in MPEG-2: the general and the restricted slice structures. In the general slice structure, MPEG-2 slices need not cover the entire picture. Thus, only the regions enclosed in the slices are encoded. In the restricted slice structure, every macroblock in the picture shall be enclosed in a slice.

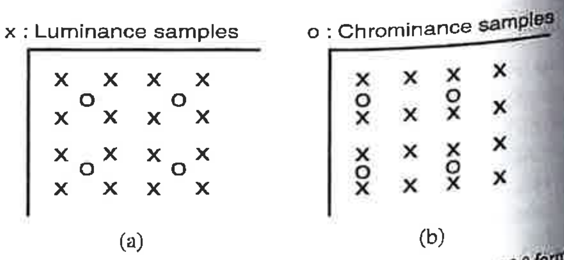


FIGURE 9 Position of luminance and chrominance samples for 4:2:0 format in (a) MPEG-1, (b) MPEG-2.

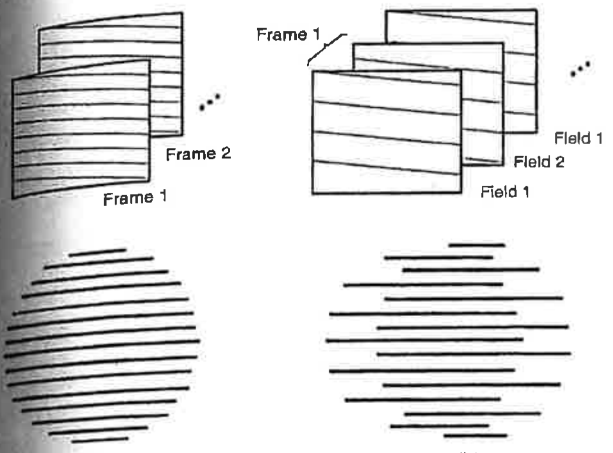


FIGURE 10 (a) Progressive scan, (b) interlaced scan.

2.4 MPEG-2 Video Coding Standard Compared with MPEG-1

2.4.1 Interlaced Versus Progressive Video

Figure 10 shows the progressive and interlaced video scan. In the interlaced video, each displayed frame consists of two interlaced fields. For example, frame 1 consists of field 1 and field 2, with the scanning lines in field 1 located between the lines of field 2. In contrast, the progressive video has all the lines of a picture displayed in one frame. There are no fields or half-pictures as with the interlaced scan. Thus, progressive video requires a higher picture rate than the frame rate of an interlaced video, to avoid a flickery display. The main disadvantage of the interlaced format is that when there are object movements, the moving object may appear distorted when we merge two fields into a frame. For example, Fig. 10 shows a moving ball. In the interlaced format, because the moving ball will be at different locations in the two fields, when we put the two fields into a frame, the ball will look distorted. Using MPEG-1 to encode the distorted objects in the frames of the interlaced video will not produce the optimal results. Interlaced video also tends to cause horizontal picture details to dither and thus introduces more high-frequency noises.

2.4.2 Interlaced Video Coding

Figure 11 shows the interlaced video format. As explained earlier, an interlaced frame is composed of two fields. From the figure, the top field (field 1) occurs earlier in time than the bottom field (field 2). Both fields together form a frame. In MPEG-2, pictures are coded as I, P, and B pictures, like in MPEG-1. To optimally encode the interlaced video, MPEG-2 can encode a picture either as a field picture or a frame picture. In the field-picture mode, the two fields in the frame are encoded separately. If the first field in a picture is an I picture, the second field in the picture can be either I or P pictures, as the second field can use the first field as a reference picture. However, if the first field in a picture is a P- or B-field picture, the second field has to be the same type of picture. In a frame picture, two fields are interleaved into a picture and coded together as one picture, similar to the conventional coding of progressive video pictures. In MPEG-2, a video sequence is a collection of frame pictures and field pictures.

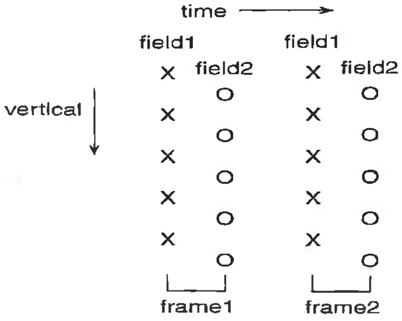


FIGURE 11 Interlaced video format.

2.4.2.1 Frame-Based and Field-Based Motion-Compensated Prediction. In MPEG-2, an interlaced picture can be encoded as a frame picture or as field pictures. MPEG-2 defines two different motion-compensated prediction types: frame-based and field-based motion-compensated prediction. Frame-based prediction forms a prediction based on the reference frames. Field-based prediction is made based on reference fields. For the simple profile in which the bidirectional prediction cannot be used, MPEG-2 introduced a dual-prime motion-compensated prediction to efficiently explore the temporal redundancies between fields. Figure 12 shows three types of motion-compensated prediction. Note that all motion vectors in MPEG-2 are specified with a half-pixel resolution.

Frame predictions in frame pictures: in the frame-based prediction for frame pictures, as shown in Fig. 12(a), the whole interlaced frame is considered as a single picture. It uses the same motion-compensated predictive coding method used in MPEG-1. Each 16 × 16 macroblock can have only one motion vector for each forward or backward prediction. Two motion vectors are allowed in the case of the bidirectional prediction.

Field prediction in a frame pictures: the field-based prediction in frame pictures considers each frame picture as two separate

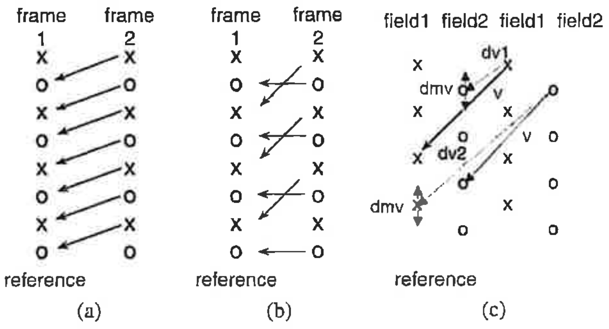


FIGURE 12 Three types of motion-compensated prediction: (a) frame, (b) field, (c) dual prime.

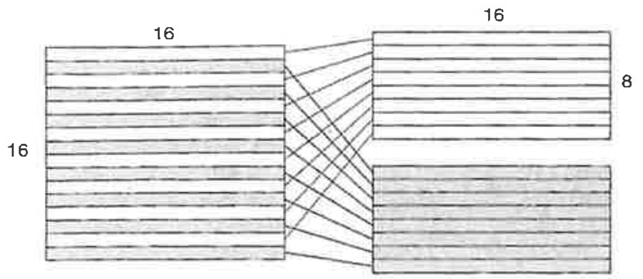


FIGURE 13 Blocks for frame-based or field-based prediction.

field pictures. Separate predictions are formed for each 16 × 8 block of the macroblock as shown in Fig. 13. Thus, field-based prediction in a frame picture requires two sets of motion vectors. A total of four motion vectors is allowed in case of bidirectional prediction. Each field prediction may select either the field 1 or the field 2 of the reference frame.

Field prediction in field pictures: in field-based prediction for field pictures, the prediction is formed from the two most recently decoded fields. The predictions are made from reference fields, independently for each field, with each field considered as an independent picture. The block size of prediction is 16 × 16; however, it should be noted that the 16 × 16 block in the field picture corresponds to a 16 × 32 pixel area in the frame picture. A field-based prediction in field pictures requires only one motion vector for each forward or backward prediction. Two motion vectors are allowed in the case of the bidirectional prediction.

16 × 8 prediction in field pictures: two motion vectors are used for each macroblock. The first motion vector is applied to the 16 × 8 block in field 1 and the second motion vector is applied to the 16 × 8 block in field 2. A total of four motion vectors is allowed in the case of bidirectional prediction.

Dual-prime motion-compensated prediction can be used only in P pictures. Once the motion vector "v" for a macroblock in a field of given parity (field 1 or field 2) is known relative to a reference field of the same parity, it is extrapolated or interpolated to obtain a prediction of the motion vector for the opposite parity reference field. In addition, a small correction is also made to the vertical component of the motion vectors to reflect the vertical shift between lines of the field 1 and field 2. These derived motion vectors are denoted by **dv1** and **dv2** (represented by dash line) in Fig. 12(c). Next, a small refinement differential motion vector, called "**dmv**", is added. The choice of **dmv** values (−1, 0, +1) is determined by the encoder. The motion vector **v** and its corresponding **dmv** value are included in the bit stream so that the decoder can also derive **dv1** and **dv2**. In calculating the pixel values of the prediction, the motion-compensated predictions from the two reference fields are averaged, which tends to reduce the noise in the data. Dual-prime prediction is mainly for low-delay coding applications such as videophone and video conferencing. For low-delay coding using simple profile, B pictures should not be used. Without using bidirectional prediction, dual-prime prediction is developed for P pictures to provide a better prediction than the forward prediction.

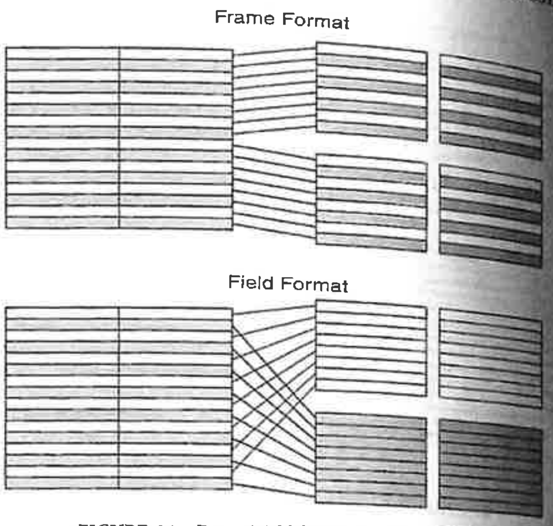


FIGURE 14 Frame/Field format block for DCT.

2.4.2.2 Frame/Field DCT. MPEG-2 has two DCT modes: frame-based and field-based DCT, as shown in Fig. 14. In the frame-based DCT mode, a 16 × 16-pixel macroblock is divided into four 8 × 8 DCT blocks. This mode is suitable for the blocks in the background or in a still image that have little motion because these blocks have high correlation between pixel values from adjacent scan lines. In the field-based DCT mode, a macroblock is divided into four DCT blocks where the pixels from the same field are grouped together into one block. This mode is suitable for the blocks that have motion because, as explained, motion causes distortion and may introduce high-frequency noises into the interlaced frame.

2.4.2.3 Alternate Scan. MPEG-2 defines two different zigzag scanning orders zigzag and alternate scans as shown in Fig. 15. The zigzag scan used in MPEG-1 is suitable for progressive images where the frequency components have equal importance in each horizontal and vertical direction. In MPEG-2, an alternate scan is introduced based on the fact that interlaced images tend to have higher frequency components in the vertical direction. Thus, the scanning order weighs more on the higher vertical frequencies than the same horizontal frequencies. In MPEG-2, the selection between these two zigzag scan orders can be made on a picture basis.

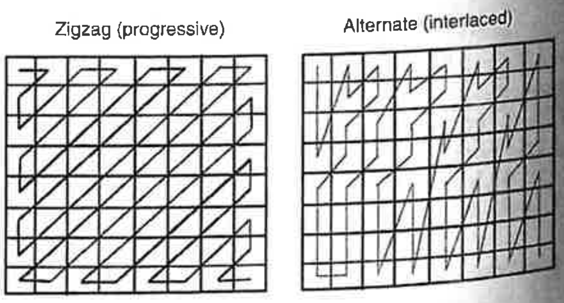


FIGURE 15 Progressive/interlaced scan.

2.5 Scalable Coding

Scalable coding is also called layered coding. In scalable coding, the video is coded in a base layer and several enhancement layers. If only the base layer is decoded, basic video quality can be obtained. If the enhancement layers are also decoded, enhanced video quality (e.g., higher SNR, higher resolution, higher frame rate) can be achieved. Scalable coding is useful for transmission over noisy channel since the more important layers (e.g., the base layer) can be better protected and sent over a channel with better error performance. Scalable coding is also used in video transport over variable-bit-rate channels. When the channel bandwidth is reduced, the less important enhancement layers will not be transmitted. It is also useful for progressive transmission, which means the users can get rough representations of the video fast with the base layer and then the video quality will be refined as more enhancement data arrive. Progress transmission is useful for database browsing and image transmission over the Internet.

MPEG-2 supports three types of scalability modes: SNR, spatial, and temporal scalability. Each of them is targeted at several applications with particular requirements. Different scalable modes can be combined into hybrid coding schemes such as hybrid spatial-temporal and hybrid spatial-SNR scalability. In a basic MPEG-2 scalability mode, there can be two layers of video: lower and enhancement layers. The hybrid scalability allows up to three layers.

2.5.1 SNR Scalability

MPEG-2 SNR scalability provides two different video qualities from a single video source while maintaining the same spatial and temporal resolutions. A block diagram of the two-layer SNR scalable encoder and decoder is shown in Figs. 16(a) and 16(b), respectively. In the base layer, the DCT coefficients are coarsely quantized and the coded bit stream is transmitted with moderate quality at a lower bit rate. In the enhancement layer, the difference between the nonquantized DCT coefficients and the coarsely quantized DCT coefficients from the lower layer is encoded with finer quantization step sizes. By doing this, the moderate video quality can be achieved by decoding only the lower-layer bit streams while the higher video quality can be achieved by decoding both layers.

2.5.2 Spatial Scalability

With Spatial scalability, the applications can support users with different resolution terminals. For example, the compatibility between SDTV (Standard Definition TV) and HDTV can be achieved with the SDTV being coded as the base layer. With the enhancement layer, the overall bit stream can provide the HDTV resolution. The input to the base layer usually is created by downsampling the original video to create a low-resolution video for providing the basic spatial resolution. The choice of video formats such as frame sizes, frame rate, or chrominance formats is flexible in each layer.

A block diagram of the two-layer spatial scalable encoder and decoder is shown in Figs. 17(a) and 17(b), respectively. In the base layer, the input video signal is downsampled by spatial decimation. To generate a prediction for the enhancement layer video signal input, the decoded lower layer video signal is upsampled by spatial interpolation and is weighted and combined with the motion-compensated prediction from the enhancement layer. The selection of weights is done on a macroblock basis and the selection information is sent as a part of the enhancement-layer bit stream.

The base- and enhancement-layer coded bit streams are then transmitted over the channel. At the decoder, the lower-layer bit streams are decoded to obtain the lower-resolution video. The lower-resolution video is interpolated and then weighted and added to the motion-compensated prediction from the enhancement layer. In the MPEG-2 video standard, the spatial interpolator is defined as a linear interpolation or a simple averaging for missing samples.

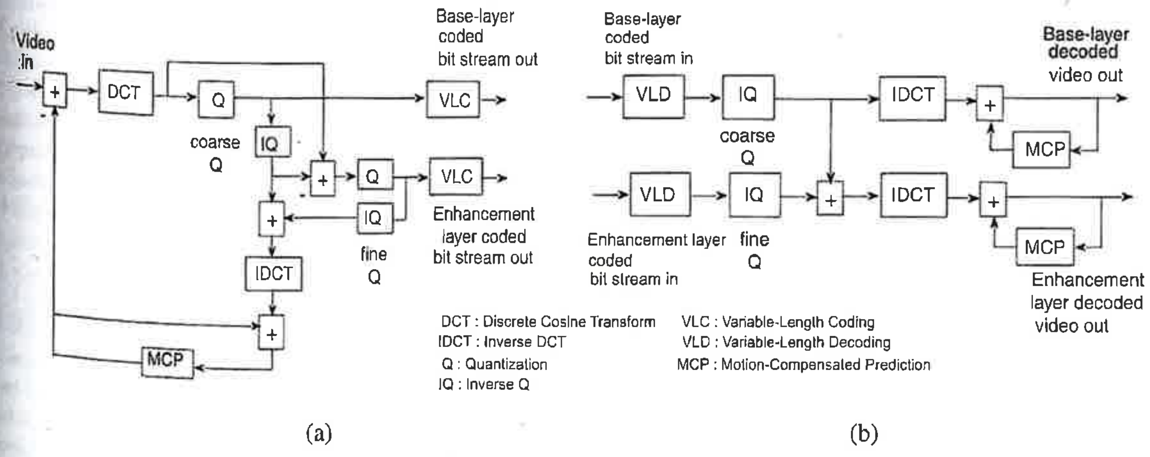


FIGURE 16 SNR scalable (a) encoder, (b) decoder.

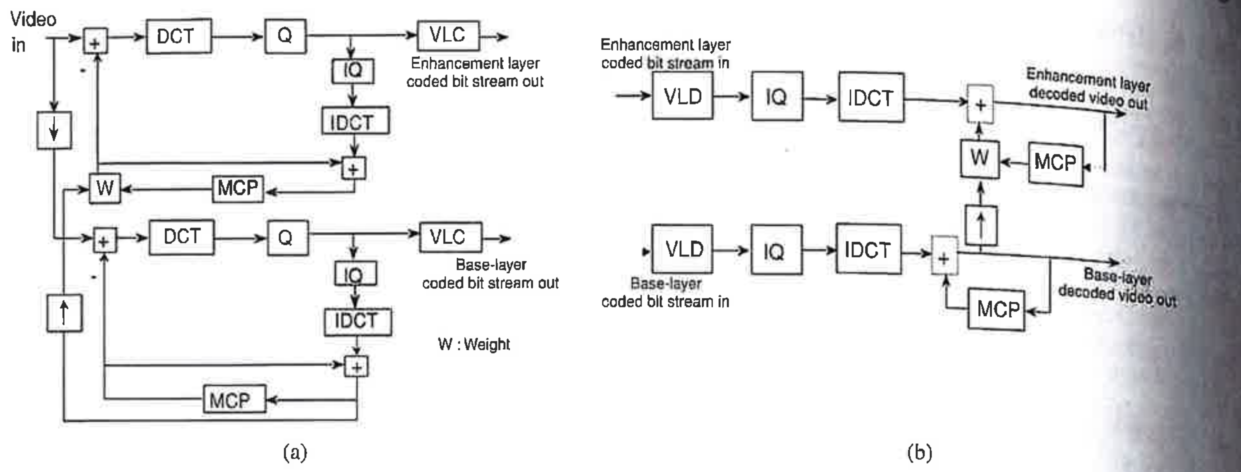

Video in
DCT
Q
VLC
IQ
IDCT
Enhancement layer coded bit stream out
W
MCP
DCT
Q
VLC
IQ
IDCT
Base-layer coded bit stream out
W : Weight
MCP
(a)
Enhancement layer coded bit stream in
VLD
IQ
IDCT
Enhancement layer decoded video out
W
MCP
VLD
IQ
IDCT
Base-layer coded bit stream in
MCP
Base-layer decoded video out
(b)

FIGURE 17 Spatial scalable (a) encoder, (b) decoder.

2.5.3 Temporal Scalability

The temporal scalability is designed for video services that require different temporal resolutions or frame-rates. The target applications include video over wireless channel where the video frame rate may need to be dropped when the channel condition is poor. It is also intended for stereoscopic video and coding of future HDTV format in which the baseline is to make the migration from the lower temporal resolution systems to the higher temporal resolution systems possible. In temporal scalable coding, the base layer is coded at a lower frame rate. The decoded base-layer pictures provide motion-compensated predictions for encoding the enhancement layer.

2.5.4 Hybrid Scalability

Two different scalable modes from the three scalability types, SNR, spatial, and temporal, can be combined into hybrid scalable coding schemes. Thus, this results in three combinations: hybrid of SNR and spatial, hybrid of spatial and temporal, and hybrid of SNR and temporal. Hybrid scalability supports up to three layers: the base layer, enhancement layer 1, and enhancement layer 2. The first combination, hybrid of SNR and spatial scalabilities, is targeted at applications such as HDTV/SDTV or SDTV/videophone at two different quality levels. The second combination, hybrid spatial and temporal scalability, can be used for applications such as high temporal resolution progressive HDTV with basic interlaced HDTV and SDTV. The last combination, hybrid SNR and temporal scalable mode, can be used for applications such as enhanced progressive HDTV with basic progressive HDTV at two different quality levels.

2.6 Data Partitioning

Data partitioning is designed to provide more robust transmission in an error-prone environment. Data partitioning splits the block of 64 quantized transform coefficients into partitions. The lower partitions contain more critical information, such as low-frequency DCT coefficients. To provide more robust transmission, the lower partitions should be better protected or transmitted with a high priority channel with a low probability of error, while the upper partitions can be transmitted with a lower priority. This scheme has not been formally standardized in MPEG-2 but was specified in the information annex of the MPEG-2 DIS document [7]. One thing to note is that the partitioned data are not backward compatible with other MPEG-2 bit streams. Therefore, it requires a decoder that supports the decoding of data partitioning. Using the scalable coding and data partitioning may result in mismatch of reconstructed pictures in the encoder and the decoder and thus cause drift in video quality. In MPEG-2, since there are I pictures that can terminate error propagation, depending on the application requirements, it may not be a severe problem.

2.7 Other Tools for Error Resilience

The effect of bit errors in MPEG-2 coded sequences varies depending on the location of the errors in the bit stream. Errors occurring in the sequence header, picture header, and slice header can make it impossible for the decoder to decode the sequence, the picture, or the slice. Errors in the slice data that contains important information such as macroblock header, DCT coefficients, and motion vectors can cause the decoder to lose synchronization or cause spatial and temporal error propagation. There are several techniques to reduce the effects of errors besides the scalable coding. These include concealment motion vectors, the slice structure, and temporal localization by the use of intra pictures/slices/macroblocks.

The basic idea of concealment motion vector is to transmit motion vectors with the intra macroblocks. Since the intra macroblocks are used for future prediction, they may cause severe video quality degradations if they are lost or corrupted by transmission errors. With a concealment motion vector, a decoder can use the best matching block indicated by the concealment motion vector to replace the corrupted intra macroblock. This improves the concealment performance of the decoder.

In MPEG, each slice starts with a slice header, which is a unique pattern that can be found without decoding the variable-length codes. These slice headers represent possible resynchronization markers after a transmission error. A small slice size, i.e., a smaller number of macroblocks in a slice, can be chosen to increase the frequency of synchronization points, thus reducing the effects of the spatial propagation of each error in a picture. However, this can lead to a reduction in coding efficiency as the slice-header overhead information is increased.

The temporal localization is used to minimize the extent of error propagation from picture to picture in a video sequence, e.g., by using intra coding modes. For the temporal error propagation in an MPEG video sequence, the error from an I or **P** picture will stop propagating when the next error-free I picture occurs. Therefore, increasing the number of I pictures/slices/macroblocks in the coded sequence can reduce the distortion caused by the temporal error propagation. However, more I pictures/slices/macroblocks will result in a reduction of coding efficiency, and it is more likely that errors will occur in the I pictures, which will cause error propagation.

2.8 Test Model

Similar to other video coding standards such as H.261 and MPEG-1, MPEG-2 only specifies the syntax and the decoder. Many detailed coding options are not specified. In order to have a reference MPEG-2 video quality, test models were developed in MPEG-2. The final test model of MPEG-2 is called "test model 5" (TM5) [10]. TM5 was defined only for main profile experiments. The motion-compensated prediction techniques involve frame, field, and dual-prime prediction, and have forward and backward motion vectors as in MPEG-1. The dual-prime was kept in main profile but restricted to P pictures with no intervening B pictures. A two-step search, which consists of an integer-pixel full search followed by a half-pixel search, is used for motion estimation. The mode decision (intra/inter coding) is also specified. Main profile was restricted to only two quantization matrices, the default table specified in MPEG-1 and the nonlinear quantizer tables. The traditional zigzag scan is used for inter coding while the alternate scan is used for intra coding. The rate-control algorithm in TMN5 consists of three layers operating at the GOP, the picture, and the macroblock levels. A bit allocation per picture is determined at the GOP layer and updated based on the buffer fullness and the complexity of the pictures.

2.9 MPEG-2 Video and System Bit-Stream Structures

A high-level structure of the MPEG-2 video bit stream is shown in Fig. 18. Every MPEG-2 sequence starts with a sequence header and ends with an end of sequence. MPEG-2 syntax is a superset of the MPEG-1 syntax. The MPEG-2 bit stream is based on the basic structure of MPEG-1 (refer to Fig. 8). There are two types of bit stream syntax allowed: ISO/IEC 11172-2 video sequence syntax or ISO/IEC 13818-2 (MPEG-2) video sequence syntax.

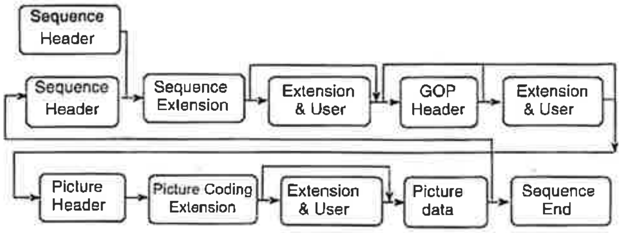


FIGURE 18 MPEG-2 data structure and syntax.

If the sequence header is not followed by the sequence extension, the MPEG-1 bit-stream syntax is used. Otherwise, the MPEG-2 syntax is used, which accommodates more features but at the expense of higher complexity. The sequence extension includes a profile/level indication, a progressive/interlaced indicator, a display extension including choices of chroma formats and horizontal/vertical display sizes, and choices of scalable modes. The GOP header is located next in the bit-stream syntax with at least one picture following each GOP header. The picture header is always followed by the picture coding extension, the optional extension and user data fields, and picture data. The picture coding extension includes several important parameters such as the indication of intra-DC precision, picture structures (choices of the first/second fields or frame pictures), intra-VLC format, alternate scan, choices of updated quantization matrix, picture display size, display size of the base layer in the case of the spatial scalability extension, and indicator of forward/backward reference picture in the base layer in the case of the temporal scalability extension. The picture data consist of slices, macroblocks, and data for the coded DCT blocks. MPEG-2 defines six layers as MPEG-1. However, the specification of some data elements is different. The details of MPEG-2 syntax specification are documented in [8].

2.10 Summary

MPEG-2 is mainly targeted at general higher quality video applications at bit-rate greater than 2 Mbit/s. It is suitable for coding both progressive and interlaced video. MPEG-2 uses frame/field adaptive motion-compensated predictive coding and DCT. Dual-prime motion compensation for P pictures is used for low-delay applications with no intervening B pictures. In addition to the default quantization table, MPEG-2 defines a nonlinear quantization table with increased accuracy for small values. Alternate scan and new VLC tables are defined for DCT coefficient coding. MPEG-2 also supports compatibility and scalability with the MPEG-1 standard. MPEG-2 syntax is a superset of MPEG-1 syntax and can support a variety of rates and formats for various applications. Similar to other video coding standards, MPEG-2 defines only syntax and semantics. It does not specify every encoding options (preprocessing, motion estimation, quantizer, rate-quality control, and other coding options) and decoding options (postprocessing and error concealment) to allow continuing technology improvement and product differentiation. It is important to keep in mind that different implementations may lead to the different quality, bit rate, delay, and complexity tradeoffs with the different cost factors. An MPEG-2 encoder is much more expensive than an MPEG-2 decoder, because it has to perform many more operations (e.g., motion estimation, coding-mode decisions, and rate-control). An MPEG-2 encoder is also much more expensive than an H.261 or an MPEG-1 encoder as a result of the higher resolution and more complicated motion estimations (e.g., larger search range, frame/field bidirectional motion estimation). References [11–25] provide further information on the related MPEG-1 and MPEG-2 topics.

References

[1] ISO/IEC JTC1 CD 10918, "Digital compression and coding of continuous-tone still images," International Organization for Standardization (ISO), 1993.

[2] ITU-T Recommendation H.261, "Line transmission of non-telephone signals. Video codec for audio visual services at px64 kbits/s," March, 1993.

[3] S. Okubo, "Reference model methodology — a tool for the collaborative creation of video coding standards," *Proc. IEEE* **83**, 139–150 (1995).

[4] MPEG proposal package description, document ISO/WG8/MPEG/89-128, July, 1989.

[5] T. Hidaka and K. Ozawa, "Subjective assessment of redundancy-reduced moving images for interactive applications: test methodology and report," *Signal Process. Image Commun.* **2**, 201–219 (1990).

[6] ISO/IEC JTC1 CD 11172, "Coding of moving pictures and associated audio for digital storage media up to 1.5 Mbits/s," International Organization for Standardization (ISO), 1992.

[7] ISO/IEC JTC1/SC2/WG11, "MPEG video simulation model three (SM3)," MPEG 90/041, July, 1990.

[8] ISO/IEC JTC1 CD 13818, "Information technology — generic coding of moving pictures and associated audio information," International Organization for Standardization (ISO), 1994.

[9] ISO/IEC 13818-2-ITU-T Rec. H.262, "Generic coding of moving pictures and associated audio information: video," 1995.

[10] ISO/IEC JTC1/SC29/WG11, "Test model 5," MPEG 93/457, document AVC-491, April, 1993.

[11] M. L. Liou, "Visual telephony as an ISDN application," *IEEE Commun. Mag.* **28**, 30–38 (1990).

[12] A. Tabatabai, M. Mills, and M. L. Liou, "A review of CCITT px64 kbps video coding and related standards," in *Proceedings of International Electronic Imaging Exposition and Conference* (Boston, MA, 1990), pp. 58–61.

[13] D. J. Le Gall, "MPEG: A video compression standard for multimedia applications," *Commun. ACM*, **34**, 47–58 (1991).

[14] D. J. Le Gall, "The MPEG video compression algorithm," *Signal Process. Image Commun.* **4**, 129–140 (1992).

[15] L. Chiariglione, "Standardization of moving picture coding for interactive applications," in *Proceedings of IEEE GLOBECOM'89*, (Dallas, TX, 1989), pp. 559–563.

[16] A. Puri, *Video Coding Using the MPEG-1 Compression Standard* (Society for Information Display International Symposium, Boston, MA, 1992), pp. 123–126.

[17] A. Puri, "Video coding using the MPEG-2 compression standard," in *Proceedings SPIE Intl. Conf. Visual Communications and Image Processing (VCIP'93)* (Cambridge, MA, 1993), vol. SPIE-2094, pp. 1701–1713.

[18] S. Okubo, K. McCann, and A. Lippman, "MPEG-2 requirements, profiles, and performance verification," presented at the International Workshop on HDTV'93, Ottawa, Canada, October 25–28, 1993.

[19] A. Puri, R. Aravind, and B. Haskell, "Adaptive frame/field motion compensated video coding," *Signal Process. Image Commun.* 5, 39–58 (1993).

[20] T. Naveen, C. Horne, A. Tabatabai, D. Messing, and R. Eifrig, "MPEG-2 4:2:2 profile at main level: an emerging high-quality video compression standard," in *Standards and Common Interfaces for Video Information Systems, SPIE Proceedings vol. CR60*, (Philadelphia, PA, 1995), pp. 288–308.

[21] A. Puri, R. Kollarits, and B. Haskell, "Compression of stereoscopic video using MPEG-2," in *Standards and Common Interfaces for Video Information Systems, SPIE Proceedings vol. CR60*, (Philadelphia, PA, 1995), pp. 309–334.

[22] R. J. Clarke, *Digital Compression of Still Images and Video* (Academic, New York, 1995).

[23] V. Bhaskaran and K. Konstantinides, *Image and Video Compression Standards: Algorithms and Architectures* (Kluwer, Boston, MA, 1995).

[24] J. L. Mitchell, W. B. Pennebaker, and D. J. Le Gall, *The MPEG Digital Video Compression Standard* (Van Nostrand Reinhold, New York, 1996).

[25] K. R. Rao and J. J. Hwang, *Techniques and Standards for Image, Video, and Audio Coding* (Prentice-Hall, Englewood Cliffs, NJ, 1996).

6.5

Emerging MPEG Standards: MPEG-4 and MPEG-7*

Berna Erol,
Adriana Dumitraş,
and Faouzi Kossentini
University of British Columbia

1 Introduction

During the past two decades, we have witnessed an increasing number of multimedia applications and services in many areas, including entertainment, education, and medicine. Multimedia technologies improve interpersonal communication, promote faster understanding of complex ideas, provide increased access capabilities to information, and allow higher interactivity levels with the media.

The vast amount of digital data that are associated with multimedia applications, and the complex interactions between the different types of data, such as text, speech, music, images, graphics, and video, make the representation, exchange, storage, access, and manipulation of these data a challenging task. In order to provide interoperability between different multimedia applications and promote further use of multimedia data, there is a need to standardize the representation of, and access to, these data. There has already been significant work in the fields of efficient representation by means of compression, storage, and transmission [1–4]. However, there has been little emphasis on the content accessibility and manipulation. The new generation of highly interactive multimedia applications require that the users be able to access and manipulate multimedia data in both uncompressed and comprêssed forms. This has fueled several recent international standardization activities, such as those of the Moving Picture Experts Group (MPEG), officially known as Working Group 11 of the ISO/IEC JTC1/SC29 technical committee. MPEG is currently developing two emerging standards: MPEG-4, which is standardizing an object-based coded representation of multimedia data, and MPEG-7, which is standardizing a multimedia content description interface.

MPEG-4, like the MPEG-1/2 [1, 2] and ITU-T H.263/H.263+ [3, 4] standards, which are discussed in Chapters 6.4 and 6.1, respectively, offers high compression performance levels, making much more efficient the storage and transmission of audiovisual data. However, the other key objectives of MPEG-4 are to enable content-based access and provide functionalities such as error resilience, scalability, and hybrid coding of synthetic and natural data [5, 6]. On the other hand, MPEG-7 is expected to enable effective and efficient content-based access and manipulation of multimedia data, and to provide functionalities that are complementary to those of the MPEG-4 standard. With the use of an MPEG-4/MPEG-7 compliant system, it will be possible to randomly access, manipulate, and process individual objects within a scene. For example, consider the video scene given in Fig. 1. Using an MPEG-4/MPEG-7 compliant decoder, the user will be able to search for podiums that are similar to the one in the video scene, or search for fish that are similar to the one shown on the screen. The user can also search for curtains that have a texture similar to that of the background. Next, besides providing a comprehensive description of the emerging MPEG-4 and MPEG-7 visual standards, we show, through examples, how MPEG-4 and MPEG-7 will together enable

*This work was supported by the Natural Sciences and Engineering Research Council (NSERC) and the National Research Council (NRC) of Canada.

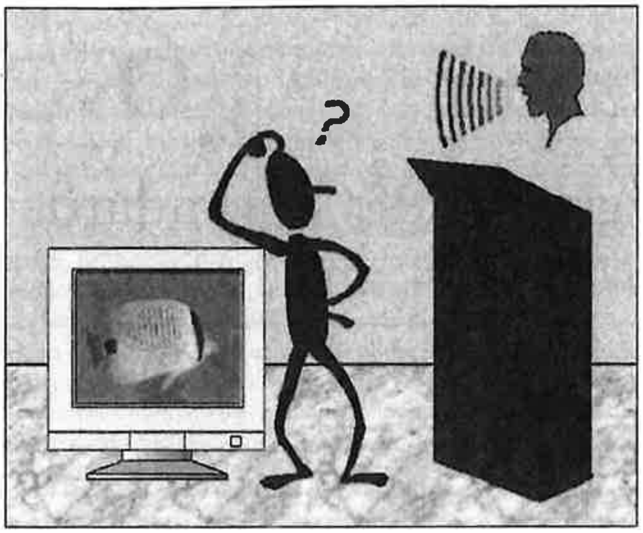


FIGURE 1 An audiovisual scene. (See color section, p. C–28.)

many desired functionalities and provide a complete multimedia solution.

2 The MPEG-4 Standard

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 the visual part [8], and the decoding of the audio data in the audio part [9]. The initial goal of MPEG-4 was to provide tools and algorithms for very low bit rate coding of audiovisual data. However, the scope has changed considerably in order to address the requirements of the new generation multimedia applications, which include multimedia communications (broadcast and interpersonal), Internet, interactive video games, video surveillance, and multimedia databases [10, 11]. Besides the need to achieve high compression performance levels, these applications require interactivity with individual objects, hybrid coding of natural and synthetic objects, and a high degree of scalability and error resilience [6, 12–14]. MPEG-4 addresses all of these requirements by providing the following functionalities: (1) improved coding efficiency by providing compression tools that are optimized for objects with a wide range of source material and bit rates, (2) object-based interactivity by enabling a high degree of user interaction with the individual audiovisual objects, (3) generic coding by providing tools for the efficient representation of both natural and synthetic objects, (4) object-based and temporal random access, (5) temporal, spatial, quality and object-based scalability, and (6) robust operation in error-prone environments.

2.1 Audiovisual Object Representation

An object-based representation is necessary to enable the above functionalities. MPEG-4 achieves object-based representation by defining audiovisual objects and coding them into separate bit stream segments [6,7,15]. An audiovisual (AV) object (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 individual AVOs can vary, such that the audio component can be (1) synthetic or natural, and (2) mono, stereo, or multichannel (e.g., surround sound), and the visual component can be natural or synthetic. Some examples of AVOs include a sound recorded with a microphone, a speech synthesized from a text, a person recorded by a video camera, and a 3-D 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. In order to allow interactivity with individual AVOs within a scene, it is essential to transmit the information that describes each AVOs 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 bit streams 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. AV objects can be organized in a hierarchical 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 the nature of their characteristics. For example, if it is a visual object, the user can zoom and rotate it. If it is an audio object, the user can change its pitch, as well as his or her listening point. Also, the quality and spatial and temporal resolutions of the individual AVOs can be modified. For example, in a mobile video telephony application, the user can request a higher frame rate and spatial resolution for the talking person than those of the background objects.

Audiovisual scenes are reconstructed and presented by audiovisual terminals at the receiver's end. As seen from Fig. 2, an audiovisual terminal receives the bit stream from a network or a storage device, demultiplexes the bit stream to retrieve elementary streams, decompresses the primitive AV objects, and finally performs composition and rendering of the reconstructed AV objects by using the corresponding scene description information. An AV terminal also manages upstream data transfer for user commands that require server-side interaction.

2.2 The MPEG-4 Visual Standard: Technical Description

The emerging MPEG-4 visual standard, officially known as ISO/IEC 14496–2 [8], aims at providing standardized core processing elements that allow efficient storage, transmission, and manipulation of visual data [16]. While the MPEG-4 visual standard, like its predecessors, defines only the bit stream syntax and

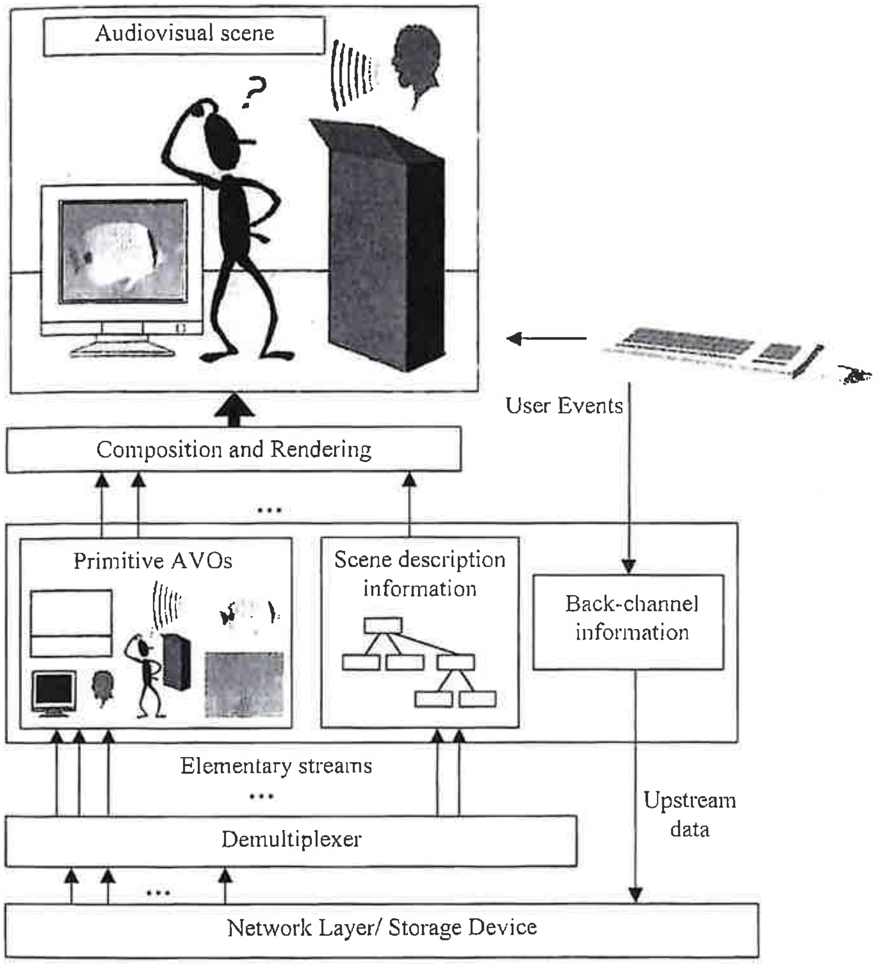


FIGURE 2 An audiovisual terminal. (See color section, p. C–29.)

the decoding process, the precise definitions of some compliant encoding algorithms are presented in two verification models: one for synthetic and natural hybrid coding (SNHC) [17], and the other one for natural video coding [18]. Although the MPEG-4 standard does not define the encoding process, both the encoding and decoding processes are discussed in this chapter.

Different representations and compression algorithms may offer optimum solutions for different applications, bit rates, and formats. Therefore, MPEG-4 provides four different types of coding tools: *Video object coding* for the coding of a naturally or synthetically originated, rectangular, or arbitrarily shaped video object; *mesh object coding* for the coding of a visual object represented with a mesh structure; *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 each of the MPEG-4 visual object coding tools. Next we discuss the scalability and the error resilience tools, followed by a presentation of the applications and profiles of the MPEG-4 visual standard. Finally, we provide an example that illustrates how MPEG-4 can be used for the coding of rectangular and arbitrarily shaped video objects.

2.2.1 Video Object Coding

A video object (VO) is an arbitrarily shaped video segment that has a semantic meaning. A 2-D snapshot of a VO at a particular time instant is called a video object plane (VOP). A VOP is defined by its texture (luminance and chrominance values) and its shape. MPEG-4 allows content-based access to not only the video objects, but also temporal instances of the video objects, i.e., VOPs. In general, MPEG-4 coding of a VOP involves coding of motion, texture, and shape information. However, 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 visual terminal must be able to decode all the bit streams of H.263 baseline encoders.

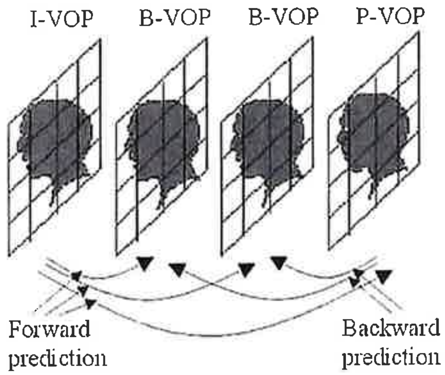


FIGURE 3 VOP prediction types.

To enable access to an arbitrarily shaped object, such an object has to be separated from the background and the other objects. This process is called segmentation, and it can be performed in real time during encoding (on line), or in nonreal time prior to encoding (off line). The segmentation process is not standardized in MPEG-4. However, there are a number of automatic and semiautomatic tools available for segmentation [19]. Also, it is possible to generate image sequences that are segmented initially by using techniques such as chroma keying [20], in which a unique color is used to separate the background from a video object.

MPEG-4 video object coding consists of shape coding (for arbitrarily shaped VOs), motion compensated prediction to reduce temporal redundancies, and DCT-based texture coding of the motion compensated prediction error data to reduce spatial redundancies. The video coding is performed at the macroblock level. VOPs are divided into macroblocks, such that they are represented with the minimum number of macroblocks within a bounding rectangle. Similar to MPEG-1 and MPEG-2, MPEG-4 supports intracoded (I), temporally predicted (P), and bidirectionally predicted (B) VOPs, all of which are illustrated in Fig. 3.

Figure 4 shows the basic VOP encoder structure. The encoder consists mainly of two parts: a hybrid of a motion compensated predictor and a DCT-based coder, and a shape coder. In the first part, motion estimation and compensation are performed (except for I-VOPs) on texture data, followed by DCT and quantization. Then, the difference between the predicted data and the original texture data is coded by variable length coding (VLC). Motion information is also encoded by using VLC. Then, the VOP is reconstructed as in the decoder, that is, by applying inverse quantization, applying inverse DCT (IDCT), and adding the resulting data to the motion compensated prediction data. The resulting VOP is then used for the prediction of future VOPs. The shape coder encodes the binary shape and the transparency information of the object. Since the shape of a VOP may not change significantly between consecutive VOPs, predictive coding is employed to reduce temporal redundancies. Thus, motion estimation and compensation are also performed for the shape of the object. Finally, motion, texture, and shape information is multiplexed with the headers to form the coded VOP bit stream. At the decoder end, the VOP is reconstructed by combining motion, texture, and shape data decoded from the bit stream.

2.2.1.1 Motion Vector Coding. In the bit stream, the motion data are transmitted in the form of motion vectors (MVs). MVs are predicted by using a spatial neighborhood of three MVs, and the prediction error is variable length coded. Motion vectors are transmitted only for P-VOPs and B-VOPs. MPEG-4 employs some advanced motion compensation techniques, such as

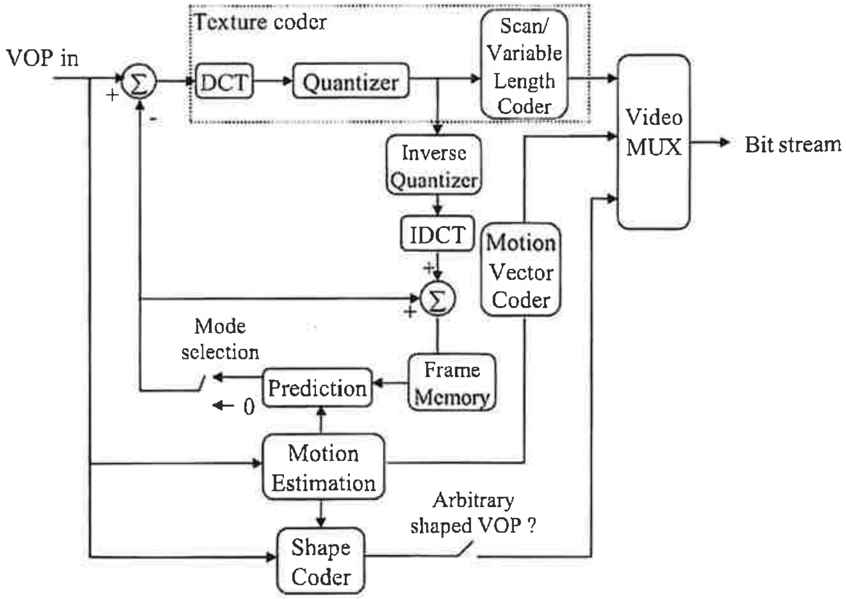


FIGURE 4 Basic block diagram of an MPEG-4 video coder.

the use of unrestricted MVs, where MVs are allowed to point outside the coded area of a reference VOP, overlapped motion compensation, and the use of four MVs per macroblock.

Since the VOPs are, in general, arbitrarily shaped, there may not be a corresponding pixel available for the prediction of the current VOP. In order to guarantee that every pixel of the current VOP can be predicted, some or all of the boundary and outside blocks of the reference VOP have to be padded by extrapolation. The boundary blocks are padded by first repeating the boundary pixels in the horizontal direction, and then repeating the boundary pixels in the vertical direction while averaging pixels whose values were obtained by horizontal padding. When a reference pixel belongs to a block that is completely outside the VOP, then the block is filled by extended padding, where pixels are assigned average values that are determined by the neighboring blocks.

2.2.1.2 ***Texture Coding.*** Intrablocks, as well as motion compensation prediction error blocks, are texture coded. Similar to MPEG-1/MPEG-2 and H.263 (described in Chapters 6.4 and 6.1, respectively), DCT-based coding is employed to reduce spatial redundancies. That is, each VOP is divided into macroblocks as illustrated in Fig. 5, and DCT coding is applied to the four 8 × 8 luminance and two 8 × 8 chrominance blocks of the macroblocks. If a macroblock lies on the boundary of an arbitrarily shaped VOP, then the pixels that are outside the VOP are padded before DCT coding. For intra-VOP boundary macroblocks, padding is performed as described in the previous section, whereas for residual blocks, the region that is outside the VOP is padded with zeros. Alternatively, a shape-adaptive DCT (SA-DCT) coder can be used to encode only those pixels that belong to the VOP. This generally results in higher compression performance, but at the expense of an increased implementation complexity. Macroblocks that are completely inside the VOP are DCT transformed as in MPEG-1/MPEG-2 and H.263. The blocks that do not belong to the VOP are not coded. DCT transformation of the blocks is followed by quantization, zigzag scanning, and variable length coding. Note that adaptive DC/AC prediction methods and alternate scan techniques can be employed for efficient coding of the DCT coefficients of intra blocks.

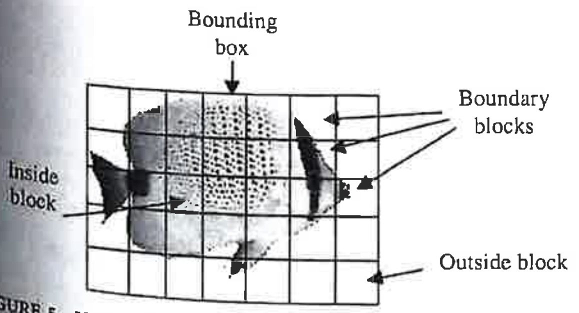


FIGURE 5 VOP enclosed in a rectangular bounding box and divided into macroblocks.

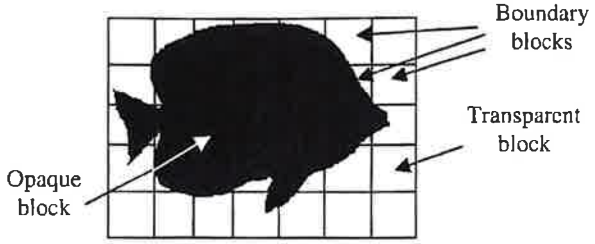


FIGURE 6 Binary alpha plane.

2.2.1.3 ***Shape Coding.*** MPEG-4 supports coding of shape information to enable content-based access to individual video objects in a scene [6, 8, 18, 20]. MPEG-4 is the only video coding standard that supports shape coding, besides H.263+ [4], which provides some limited shape coding support by means of its chroma-keying coding technique. Because of its limitations on shape rate control and its unstable performance for complex shapes, the chroma-keying coding technique was not considered for shape coding in MPEG-4 [20]. Polygon-based and bitmap-based shape coding techniques were found to be better candidates. Because of its high compression performance and low complexity, a bitmap-based shape coder was adopted.

In bitmap-based shape coding, the shape and transparency of a VOP are defined by their binary and gray-scale (respectively) alpha planes. A binary alpha plane indicates whether or not a pixel belongs to a VOP. A gray-scale alpha plane indicates the transparency of each pixel within a VOP. MPEG-4 provides tools for both lossless and lossy coding of binary and gray-scale alpha planes. Furthermore, both intra shape and inter shape coding are supported.

Binary alpha planes are divided into 16 × 16 blocks, as illustrated in Fig. 6. 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 pixels in boundary blocks (i.e., blocks that contain pixels both inside and outside the VOP) are scanned in a raster scan order and coded by using context-based arithmetic coding. In intracoding, a context is computed for each pixel using 10 neighboring pixels, which are shown in Fig. 7(a), by using the equation $C = \sum_k c_k 2^k$, where k is the pixel index, and c_k is "0" for transparent pixels and "1" for opaque pixels. Pixels from neighboring blocks are used to build the context if

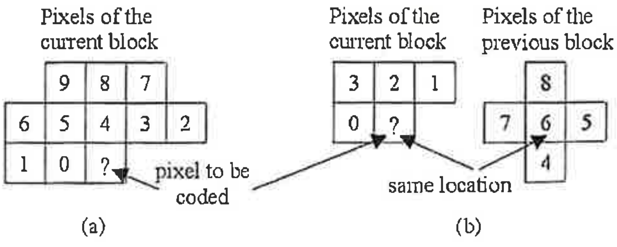

Pixels of the current block
9
8
7
6
5
4
3
2
1
0
?
pixel to be coded
(a)
Pixels of the current block
3
2
1
0
?
Pixels of the previous block
8
7
6
5
4
same location
(b)

FIGURE 7 Template pixels that form the context of arithmetic coder for (a) intracoded and (b) intercoded shape blocks.

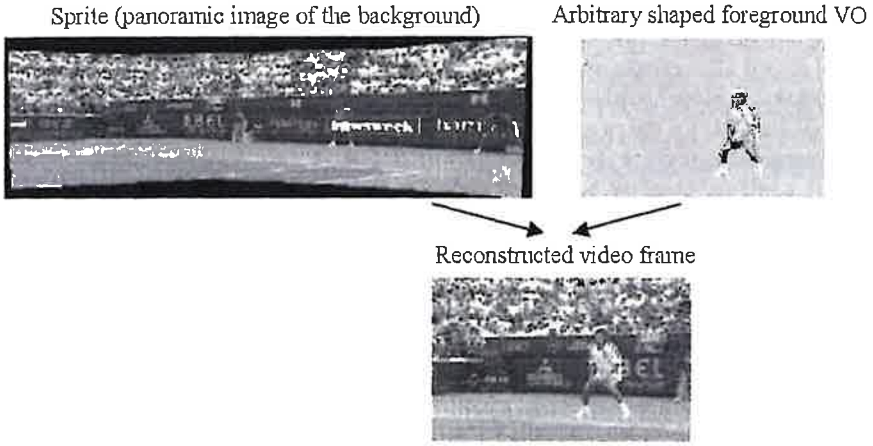

Sprite (panoramic image of the background)
Arbitrary shaped foreground VO
Reconstructed video frame

FIGURE 8 Sprite coding of a video sequence. (Courtesy of Dr. Thomas Sikora.) (See color section, p. C–30.)

the context pixels fall outside the current block. The computed context is 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 less coding bits is conveyed in the bit stream.

In inter shape coding, the shape of the current block is first predicted from the shape of the temporally previous or future VOP (depending on the VOP coding type) by performing motion estimation and compensation in integer pixel accuracy. The shape motion vector is then coded predictively. Then, the difference between the current and the predicted shape block is arithmetically coded. The context for an intercoded shape block is computed by using a template of nine pixels from both the current and temporally previous VOP shape blocks, as shown in Fig. 7(b).

Lossy coding of the binary shape is achieved by either not transmitting the difference between the current and the predicted shape block (in inter shape coding), or by subsampling the binary alpha plane by a factor of 2 or 4 prior to arithmetic encoding (in both intra- and intercoding). In order to reduce the blocky appearance of the decoded shape caused by lossy coding, an upsampling filter is employed during the reconstruction.

Transparency of pixels can take values in the range of 0 (transparent) to 255 (opaque). If all of the pixels in a VOP block are opaque or transparent, then no transparency information is transmitted for that block. Otherwise, gray-scale alpha planes, which represent transparency information, are divided into 16 × 16 blocks and coded the same way as the texture in the luminance blocks.

2.2.1.4 Sprite Coding. In MPEG-4, sprite coding is used for representation of video objects that are static throughout a video scene, or their changes can be approximated by warping the original object planes [8, 21]. Sprites are generally used for transmitting background in video sequences. They are coded in the same way as intra VOPs and are saved in a buffer at the decod reconstruct the video sequences. An example of a sprite is sho in Fig. 8. As seen here, a sprite may consist of a panoramic im of the background, including the pixels that are occluded by other video objects. Such a representation can increase cod efficiency, since the background image is coded only once at beginning of the video segment, and the camera motion, su as panning and zooming, can be represented by a few transf mation coefficients in the rest of the frames.

2.2.2 Mesh Object Coding

A mesh is a tessellation (partitioning) of an image into polygon patches. Mesh representations have been successfully used computer graphics for efficient modeling and rendering of 3- objects. In order to benefit from functionalities provided by suc representations, MPEG-4 supports 2-D mesh representations natural and synthetic visual objects, and still texture objec with triangular patches [8, 22]. The vertices of the triangul mesh elements are called node points, and they can be used track the motion of a video object, as depicted in Fig. 9. Motio compensation is performed by spatially piecewise warping the texture maps that correspond to the triangular patches. Th representation provides a good model for spatially continuou motion fields.

An initial 2-D triangular mesh can be either a uniform mes or a Delaunay mesh. An example of a uniform mesh is show

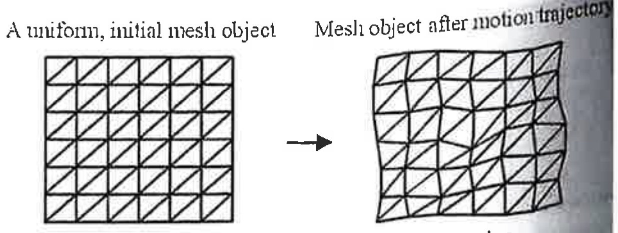

A uniform, initial mesh object
Mesh object after motion trajectory

FIGURE 9 Mesh object with triangular patches.

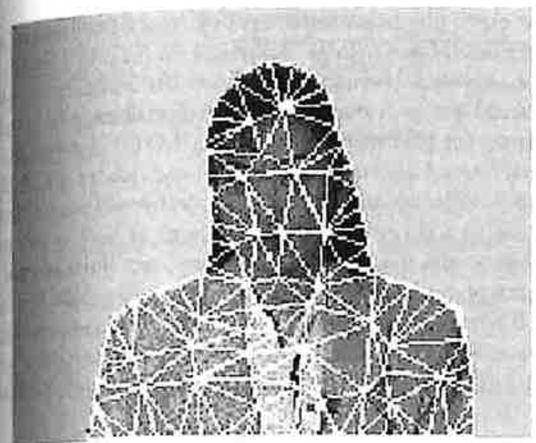


FIGURE 10 Mesh representation of a video object with triangular patches. (Courtesy of Dr. Murat Tekalp.) (See color section, p. C–30.)

in Fig. 9. A uniform mesh can be represented by a small set of parameters: the width and height of the mesh rectangle, and the type of the mesh structure. On the other hand, Delaunay meshes provide more flexibility by allowing initial node points to be at any location. The locations of the node points are coded differentially with respect to the previously coded node point coordinate. The ordering of the node points is such that the boundary node points are coded first, followed by coding of the interior node points. As seen in Fig. 10, a Delaunay mesh can be adapted to the image content for a more accurate representation of the video object. The selection process of the node points for a Delaunay mesh and the tracking of mesh node points are not specified in the MPEG-4 standard.

Similar to VOPs, instances of mesh objects are called mesh object planes (MOPs). The structure (in the case of intracoding) and motion (in the case of intercoding) of MOPs are variable length coded into a nonscalable bit stream. The texture of the corresponding visual object has to be coded separately.

2.2.2.1 *Functionalities.* A mesh-based representation of an object enables many functionalities. It improves content-based manipulation by enabling the merging of synthetic objects with natural objects. It also allows us to transmit only selected key frames, which can be animated to construct intermediate frames at the decoder. Moreover, mesh modeling can efficiently represent continuous motion, resulting in less blocking artifacts at low bit rates as compared with the block-based modeling. It also enables content-based retrieval of video objects by providing accurate object trajectory information and syntax for vertex-based object shape representation, which is more efficient than the bitmap representation.

2.2.3 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 supports the coding of two types of models [8,23,24]: a *face object* model, which is a synthetic representation of the human face with 3-D 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 3-D polygon meshes that can be rendered to simulate body movements.

2.2.3.1 Face Animation. It is required that every MPEG-4 decoder that supports face object decoding has a default face model that 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. MPEG-4 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. 11. MPEG-4 also defines high-level expressions, such as joy, sadness, fear, and surprise, and visemes for determining the mouth movements for speech animation. High-level expressions consist of a set of low-level expressions. For example, the joy expression is defined by relaxed eyebrows and open mouth, with the mouth corners pulled back toward the ears. Figure 12 illustrates several video scenes that are constructed by using face animation parameters.

The FAPs are coded by 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, improving compression efficiency, but also increasing delay.

2.2.3.2 Body Animation. 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

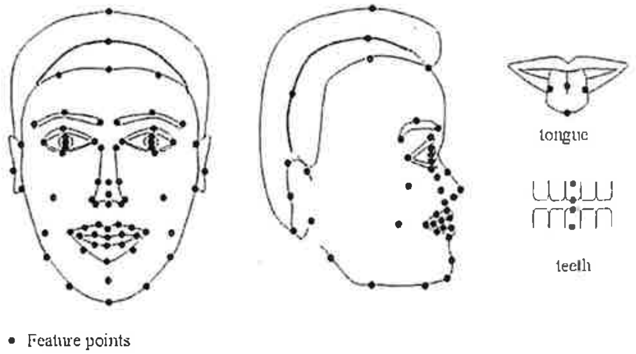


FIGURE 11 Feature points used for animation.

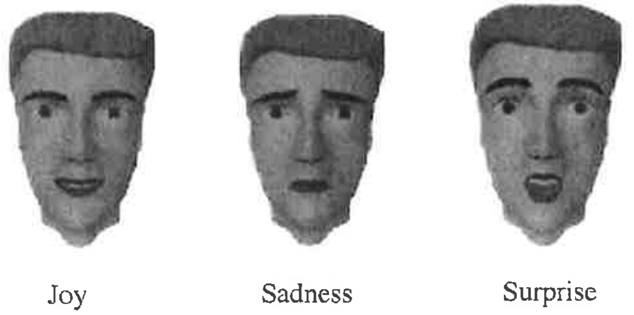


FIGURE 12 Examples of face expressions coded with FAPs. (Courtesy of Joern Ostermann.)

dimensions, surface and texture, and body animation parameters (BAPs), which define the posture and animation of a given body model. Body animation is being standardized in the Version 2 of the MPEG-4 standard.

2.2.4 Still Texture Coding

The block diagram of an MPEG-4 still texture coder is shown in Fig. 13. As depicted in the figure, the still texture is first decomposed using a 2-D separable wavelet transform, employing a Daubechies biorthogonal filter bank [8]. The discrete wavelet transform is performed using either integer or floating point operations. Also, a shape adaptive wavelet transform can be employed for coding arbitrarily shaped texture.

The DPCM coding method is applied to the coefficient values of the lowest frequency subband. A multiscale zero-tree coding method [26] is applied to the coefficients of the remaining subbands. Zero-tree modeling is used for encoding the location of nonzero 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 employed to achieve spatial and SNR scalability. After DPCM coding of the coefficients of the lowest frequency subband, and zero-tree scanning of the remaining subbands, the resulting data are coded by using an adaptive arithmetic coder.

2.2.5 Scalability

Scalability means that a bit stream consists of a separately decodable base layer, and associated enhancement layers. This structure is especially desirable for heterogeneous environments to counter limitations such as constraints on bit rate, display resolution, network throughput, and decoder complexity. Moreover, scalability provides improved error resilience by allowing the syntax for prioritized transmission. MPEG-4 supports traditional frame-based temporal, spatial, and quality scalabilities, as well as object-based scalability. Object-based scalability allows one to add or remove video objects, as well as prioritize the objects within a scene. MPEG-4 supports both spatial and temporal object-based scalability. With the use of this functionality, it is possible to represent the objects of interest with a higher spatial or temporal resolution, while allocating less bandwidth and computational power to the objects that are not as important.

2.2.6 Error resilience

MPEG-4 offers error resilience tools to address the problem of robust operation over error-prone channels. These tools can be divided into three groups: resynchronization, data partitioning, and data recovery [8, 27].

If an error occurs during the transmission of the bit stream, then resynchronization is required to recover data and conceal the effects of errors. MPEG-4 allows resynchronization by employing a method that is similar to the group of macroblocks approach of H.263 [3]. The difference is that, in order to provide periodic resynchronization markers, the number of macroblocks in an MPEG-4 packet may be variable, depending on the number of bits required to represent each macroblock. Each video packet contains information such as macroblock number and quantizer, 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 bit stream to improve the ability to localize the errors. This technique provides enhanced concealment capabilities. For example, if texture information is lost, motion information can be used to conceal the errors. Error concealment, however, is not standardized in MPEG-4.

Reversible variable length codes (RVLCs) can be employed for the coding of macroblock texture information for improved error resilience. RVLCs can be decoded in both the forward and backward directions. Thus, if part of a bit stream cannot be decoded in the forward direction because of errors, data can be recovered partially by decoding in the backward direction.

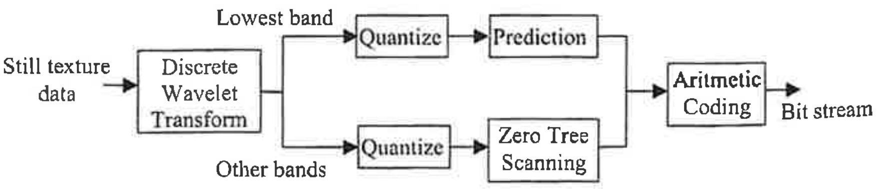


FIGURE 13 Block diagram of the still texture coder.

TABLE 1 MPEG-4 visual profiles

Profile Group	Profile Name	Supported Functionalities
Natural video	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 + transparency coding + sprite coding
	N-bit	Core profile + coding video objects with pixel depths between 4 and 12 bits
Synthetic video	Simple face animation	Basic coding of simple face animation
	Scalable texture	Spatially scalable coding of still texture objects
Hybrid (natural/synthetic) video	Simple basic animated 2-D 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

2.3 Applications and Profiles

MPEG-4 is designed to address a wide range of multimedia applications, which cover interactive video communications (e.g., video telephony and conferencing), noninteractive video communications (e.g., video e-mailing and multimedia broadcasting), digital storage media (e.g., optical disks), content-based image and video databases, video surveillance, and interactive video games. Since the MPEG-4 syntax is designed to be very generic and includes many tools to enable a wide variety of applications, the implementation of a decoder that supports the full syntax will most often be impractical. Therefore, the MPEG-4 visual standard defines a number of subsets of the syntax, referred to as "profiles" (Table 1), each targeting a specific group of applications. For example, the simple profile targets low-complexity and low-delay applications, such as mobile video communications, the main profile targets interactive broadcast and DVD applications, the *N*-bit profile targets surveillance applications, and the scalable texture profile targets applications that require multiple texture scalability levels, such as mapping texture onto objects in video games.

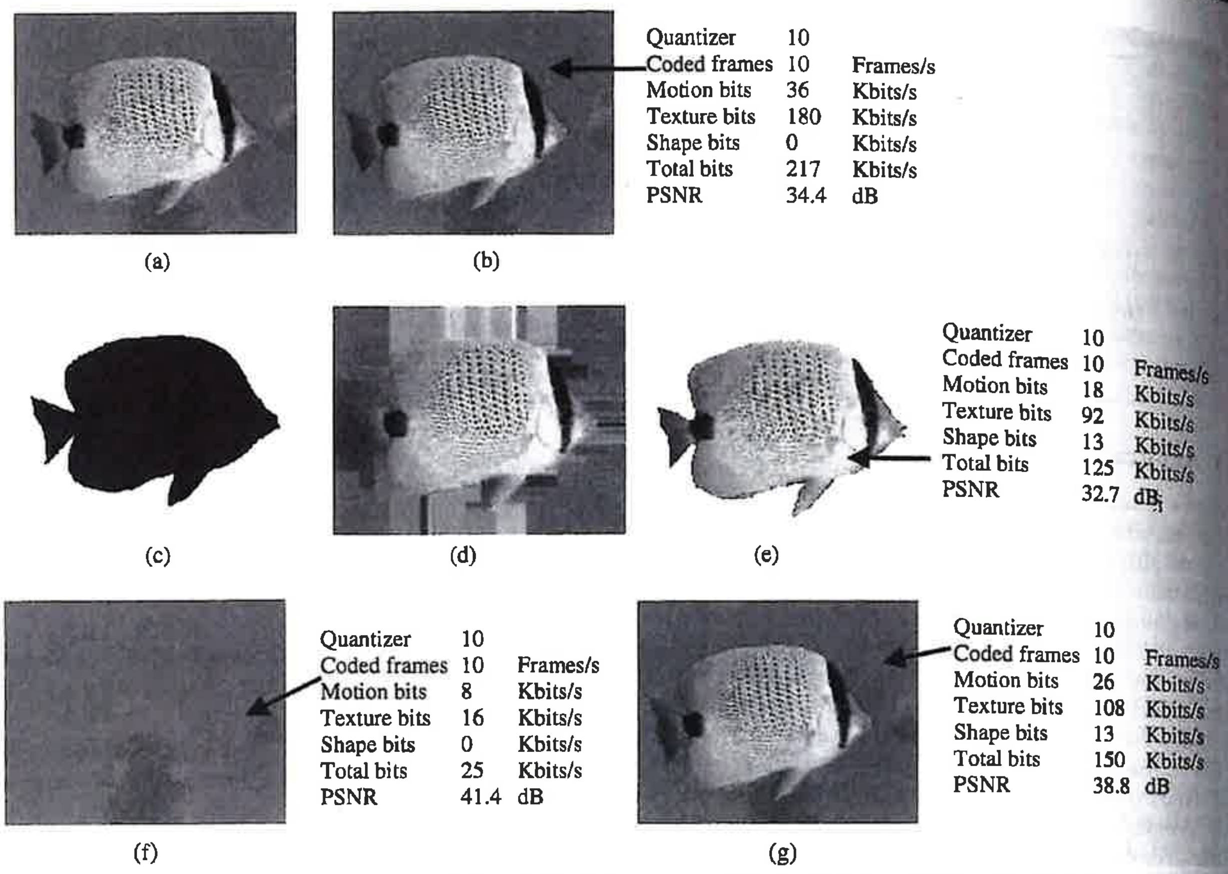
2.4 MPEG-4 Video Coding Example

In this section, we present an example to illustrate the capabilities and compression performance levels of an MPEG-4 compliant video encoder. We performed our simulations by using Microsoft's MPEG-4 encoder software [28] to encode the video sequence called *Bream*, which shows a fish that changes directions while swimming. The segmented sequence is coded following two different modes of operation that represent two distinct MPEG-4 profiles: the simple profile and the core profile. The simulation results are given in Fig. 14. The figure shows the reconstructed frames corresponding to the two used profiles, as well as the number of bits used to represent motion, texture, and shape information. In the example, the 100 frames of the *Bream* sequence are encoded at 10 frames per second (fps) and with a constant quantizer of 10. The first frame is intracoded and the rest of the frames are intercoded. In the object-based coding case (i.e., core profile), lossless shape coding is employed. Figure 14(a) shows the original input frame (the first frame of *Bream*), and Fig. 14(b) shows the reconstructed frame after using an encoder that is compliant with the simple profile (no shape coding). In this example, the simple profile coder achieves 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.

The *Bream* video sequence consists mainly of two objects: a fish (foreground object), and a water background (background object). Using the core profile encoder, we encode these two objects into two separate bit streams. Figure 14(c) shows the shape of the foreground object. We encode only the pixels that are inside the shape, which are indicated by a darker color. Texture padding of the boundary blocks is shown in Fig. 14(d). Figure 14(e) shows the foreground object as it is decoded and displayed. In this example, 10%, 14%, and 73% of the total bits are spent to represent the shape, motion vectors, and texture information, respectively. The rest of the bits are used for headers and bit stuffing. These ratios would change depending on the sequence. For example, if the shape of the sequence is changing rapidly, then more bits will be spent for shape coding.

Figure 14(f) shows the background of the sequence. The combination of background and foreground objects is shown in Fig. 14(g). A compression ratio of 80:1 is obtained. Since the background object does not vary significantly with time, the number of bits spent for its representation is very small. Here, it is also possible to employ sprite coding by selecting background as a sprite.

The PSNR versus rate performance of the frame-based and object-based coders for the 100 frames of the *Bream* video sequence is presented in Fig. 15. As seen here, 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 slowly varying foreground and background objects. However, for scenes with complex and quickly varying shapes, since a considerable amount of bits would be spent for shape coding, frame-based



Quantizer	10	
Coded frames	10	Frames/s
Motion bits	36	Kbits/s
Texture bits	180	Kbits/s
Shape bits	0	Kbits/s
Total bits	217	Kbits/s
PSNR	34.4	dB

Quantizer	10	
Coded frames	10	Frames/s
Motion bits	18	Kbits/s
Texture bits	92	Kbits/s
Shape bits	13	Kbits/s
Total bits	125	Kbits/s
PSNR	32.7	dB

Quantizer	10	
Coded frames	10	Frames/s
Motion bits	8	Kbits/s
Texture bits	16	Kbits/s
Shape bits	0	Kbits/s
Total bits	25	Kbits/s
PSNR	41.4	dB

Quantizer	10	
Coded frames	10	Frames/s
Motion bits	26	Kbits/s
Texture bits	108	Kbits/s
Shape bits	13	Kbits/s
Total bits	150	Kbits/s
PSNR	38.8	dB

FIGURE 14 Illustration of MPEG-4 coding, simple profile vs. core profile: (a) original frame; (b) frame-based coded frame; (c) shape mask for the foreground object; (d) coded foreground object (boundary macroblocks are padded); (e) foreground object as it is decoded and displayed; (f) background object as it is decoded and displayed; (g) foreground + background objects (e + f). (Bream video sequence is courtesy of Matsushita Electric Industrial Co., Ltd.) (See color section, p. C–31.)

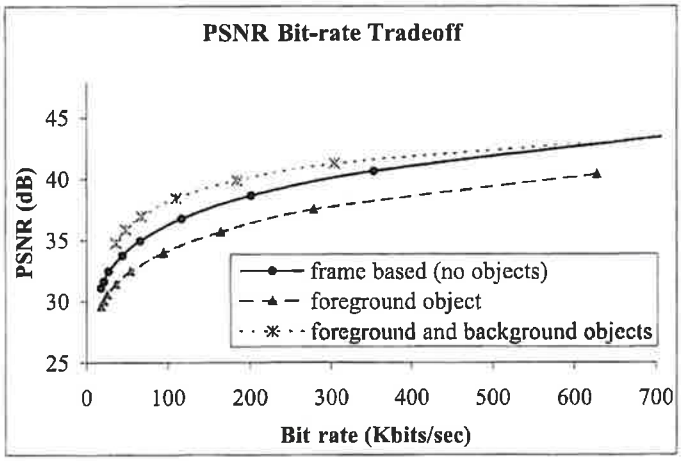


FIGURE 15 PSNR performance for the 100 frames of the *Bream* video sequence, using different profiles of the MPEG-4 video coder.

coding would achieve better compression levels, but at the cost of a limited content-based access capability.

3 The MPEG-7 Visual Standard

Many of the current multimedia applications require that the visual data be effectively and efficiently accessed and manipulated. Many text-based methods have been applied to the access and manipulation of visual content, where keywords are associated with each visual component. In order to overcome the limitations of the text-based methods, which typically require human assistance in describing visual content, feature-based methods have been introduced. Low-level features, such as texture, shape, and color, and high-level features, such as composition information, have been employed in many of the existing content-based access and manipulation (CBAM) systems. As they arise from different applications, these systems make use of various feature representations. For instance, the same "shape" feature may be represented by Fourier descriptors, geometric descriptors, etc. Therefore, data accessibility and interoperability between these systems are quite limited.

A unified framework for content representation can overcome the above problems. Moreover, such a framework would be very useful in the evaluation of current systems by various research and industry organizations, as well as for future research and development. Hence, it is not surprising that current international standardization committees, such as the MPEG committee, have focused on the standardization of a "multimedia content description interface" (MPEG-7). The major challenges facing the MPEG-7 standardization activity is that visual data can have different formats (e.g., uncompressed, compressed), different types (e.g., still pictures, audio, video), can be described by using heterogeneous feature representations, and can reside in different geographical locations.

MPEG-7 requirements for the systems, visual, and audio parts have already been developed [29, 30]. Here, we focus on the visual part of the MPEG-7 standard. We first describe the current work in the access and manipulation of visual data. Next, we present the objectives of the MPEG-7 visual standard and its normative components. Finally, we illustrate, through an example, how MPEG-7 will impact the CBAM of visual data.

3.1 Text-Based CBAM of Visual Data

Many of the text-based search methods that have been proposed are currently employed in the search engines of the World Wide Web. There are several types of text-based search engines, such as the full-text (e.g., Alta Vista, Lycos), catalogue-based (e.g., Excite, Yahoo!), meta- (e.g., Netsearch), and specialist (e.g., Bigfoot White Pages) search engines. *Full-text* search engines analyze the content of files in order to find the desired text. *Catalogue-based search* (also known as index-search) engines use classification systems in order to help the users identify the files that have been marked by human agents as being potentially useful to a particular topic. *Meta-search* (also known as multisearch) engines allow the users to search for keywords, using several search engines sequentially or simultaneously. *Specialist search* engines provide responses that are relevant to specific application areas.

TABLE 2 Examples of text-based search engines for visual content

Type	Name	Reference	Data Format
Still images	Icon Browser	33	GIF
	Image Surfer	34	JPEG
	Lycos Media	35	JPEG
	Virtual Image Archive	36	GIF, JPEG
	Yahoo Image Surfer	37	JPEG
Video	Whoopie	38	MPEG, AVI, and others
	Lycos Media	35	MOV

All of the above text-based search methods can be applied to the access and manipulation of the visual content by assigning keywords to each visual component [31, 32]. Examples of such text search engines used for CBAM of visual content are shown in Table 2. Some of the existing standards, such as HTML, provide methods to associate a text descriptor with a still image. However, HTML does not provide a mechanism for attaching other sets of descriptors to images. The SGML standard overcomes this problem. Unfortunately, the vocabulary is restricted, and similarity-based retrieval cannot be performed. Moreover, human assistance in describing the content and entering the description in the database is required.

3.2 Feature-Based CBAM of Visual Data

Feature-based methods have been proposed in order to overcome the limitations of the text-based search methods for accessing visual content. The features that are employed by the CBAM methods can be divided into two classes: low-level and high-level features [39]. The low-level features can often be extracted automatically. However, the extraction of the high-level features usually requires human assistance.

Most of the current research in content-based access and manipulation of visual data has focused on using low-level features such as texture, shape, and color [40, 41]. Texture-based CBAM of visual data has been applied in [41, 42]. These systems use texture analysis methods that are based on structural, statistical, spectral, stochastic model-based, morphology-based, or multiresolution techniques [43–46]. Shape-based CBAM methods of visual data have been proposed [47, 48] that employ various boundary-based (e.g., chain codes, geometric, and Fourier descriptors) or region-based (e.g., area, roundness) shape models. Color features have been extensively used for the CBAM of image databases [49, 50], because of their invariance with respect to image scaling and rotation. The color features have been frequently represented by computing the average color, the dominant color, and the global/local histograms [49].

In many cases, using only one low-level feature may not be sufficient to discriminate between several objects. Therefore, combinations of two or several low-level features, as employed in [39, 51–55], can improve significantly the outcome of the CBAM of visual data.

3.3 Objectives of the MPEG-7 Visual Standard

MPEG-7 is the most recent standardization activity of the MPEG group. The goal of MPEG-7 is to provide a standardized description that allows effective and efficient access and manipulation of the multimedia content [29, 30, 56]. MPEG-7 will standardize a set of descriptors (Ds), a set of description schemes (DSs), a description definition language (DDL), and schemes for the coding of the descriptions [29, 56]. MPEG-7 will 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., content recognition tools). The MPEG-7 requirements posed indirectly on the visual description tools would likely yield effective and efficient tools for segmentation, feature extraction, and visual recognition.

3.4 Visual Description

In this section, we describe the normative components, i.e., the Ds, the DSs, the DDL, and the coding schemes, of the visual part of MPEG-7.

3.4.1 Descriptors

For a given visual content (e.g., images, video), 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. Figure 16 illustrates the relationship between data, features, and descriptors. Using feature extraction, one projects the input visual 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 visual 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. Examples of descriptors associated with the input features are presented in Table 3. For instance, the shape feature may be represented by geometric descriptors or Fourier descriptors. Some of these descriptors are standardized in MPEG-7 (i.e., they belong to the standardized descriptor space). The projection from the visual data space to the feature space, which is not standardized in MPEG-7, is not unique since different applications may require different features for describing the same visual content. The projection from the feature space to the descriptor space, which is being standardized in MPEG-7, is also not unique since several descriptors may be assigned to the same feature. An MPEG-7 descriptor should be relevant and effective. This guarantees that the descriptor expresses precisely and completely the associated feature. Moreover, it should have expression and processing efficiency. This guarantees the existence of an efficient method for computing the descriptor value. Descriptor scalability with the application and with the data are also required. Finally, the descriptor should provide a multilevel representation of the associated feature. Other requirements are included in [30].

TABLE 3 Examples of features and their descriptors

Feature	Descriptor
Texture	contrast, coarseness, directionality, Markov model, Co-occurrence matrix, DCT coefficients, wavelet coefficients, Wold coefficients
Shape	geometrical descriptors (area, perimeter, etc.), Fourier descriptors, chain code
Color	color histogram, color moments
Appearance	text, Fourier coefficients

3.4.2 Description Scheme

A description scheme (DS) is the pair $\{S, R\}$, where S is the structure consisting of several components, and R is the set of relationships between the components of S. These components are descriptors, descriptors and other description schemes, or description schemes. Similar to an MPEG-7 descriptor, an MPEG-7 description scheme must be relevant and effective. Moreover, it must have expression efficiency, extensibility, and scalability with the application and with the data. DS relevance and effectiveness are guaranteed if the DS components and relationships between these components are also relevant and effective. Expression

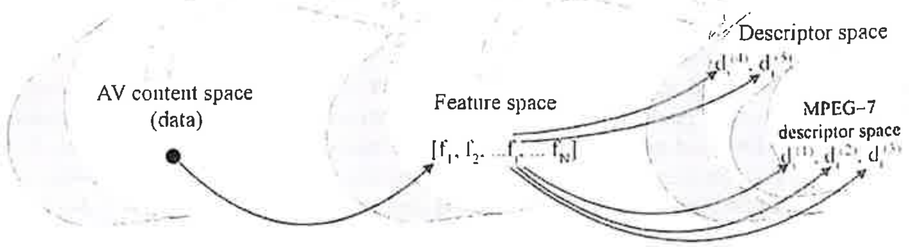


FIGURE 16 Relationship among data, features, and descriptors.

efficiency is guaranteed by obeying the parsimony principle, i.e., by employing the minimum number of DS components and relationships between these components. Finally, the DS should provide a multilevel representation of the data. Other MPEG-7 DS requirements are described in [30].

3.4.3 Description Definition Language

The description definition language (DDL) is the language used to specify the description schemes. MPEG-7 requires that the DDL be explicit by following an unambiguous grammar. Moreover, the DDL should have compositional capabilities, by allowing new DSs to be created and existing DSs to be extended. Most importantly, the DDL should be platform independent [57].

3.4.4 Coding of the Descriptions

A coded description is a representation of the description that allows efficient storage and transmission. MPEG-7 will standardize error resilient and low complexity methods for the efficient coding of the descriptions [57].

3.5 MPEG-7 Example: A Generic Visual Scene

In this section, we discuss how MPEG-7 will provide solutions to the problems associated with CBAM of visual data. Consider the audiovisual scene illustrated in Fig. 1. Suppose that we want to retrieve a picture that is similar to the one shown in the figure by submitting the same query "retrieve all the pictures containing fish" to two different CBAM systems: System A and System B. System A employs Fourier descriptors (coefficients of the Fourier transform of the fish boundary) for shape feature representation. Therefore, it will process the query by retrieving all the objects in the library having similar (according to a specific similarity measure) Fourier descriptors to those of the fish extracted from the query. System B uses geometric descriptors (e.g., area, perimeter) for shape feature representation, and it will then process the query by retrieving all the objects in the library having similar geometric descriptors to those extracted from the fish. The response of the systems A and B to the submitted query will clearly be different, even if both systems were to access the same digital library. This is due to the following two reasons. First, the query may be processed differently by these systems. For example, System A may accept sketch-based queries, whereas System B may accept picture-based queries. Second, even if the query were to be processed in an identical manner, the different shape feature representations and different similarity measures, would most definitely yield different results. This will likely pose problems for most of today's CBAM applications.

MPEG-7 addresses the above problems by providing a standardized description interface. That is, if System A and System B were MPEG-7 compliant, the shape feature representation would be the same in the sense that the two systems would use the same descriptors. Moreover, an MPEG-7 compliant system would achieve a better retrieval performance level than that of existing CBAM systems because of the following reasons. First, MPEG-7 would attach descriptors only to the relevant features. For instance, no descriptors would be attached to the texture feature for the black character shown in Fig. 1. Second, in an MPEG-7 description, the relevant features would be prioritized. For example, a higher importance level would be assigned to the shape descriptors than to the color descriptors for the fish. Finally, MPEG-7 would provide a hierarchical description of the audiovisual scene, as illustrated in Fig. 17. This would allow for coarse to fine representations of the audiovisual content and improve the description's accuracy.

4 Conclusions: Towards a Complete Multimedia Solution

In this chapter, we have presented a comprehensive technical description of the visual parts of the two emerging MPEG standards: MPEG-4 and MPEG-7. We showed, through examples, how these standards will enable many desired functionalities, such as efficient content-based representation, access, and manipulation of multimedia data, which are not addressed properly by today's multimedia standards.

MPEG-4 becomes an international standard in January 1999. A second version of MPEG-4, which will be backward compatible with the first version and will feature more functionalities and profiles, is expected to be completed by the end of 1999. The work of MPEG-7, however, is still in its infancy. In fact, the MPEG-7 call for proposals has just been issued (October 1998). MPEG-7 is expected to become an international standard in September 2001 [29].

MPEG-4 achieves high compression levels, making efficient the communication of multimedia content. Through its object-based representation and modeling tools (e.g., mesh, sprite), MPEG-4 allows us to combine graphics, text, and synthetic/natural objects in a single bit stream. MPEG-4 also features scalability and error resilience functionalities enabling efficient and robust transmission of multimedia data. MPEG-7 will build on MPEG-4, making use of the object-based representation and modeling tools, and providing complementary functionalities. MPEG-7 will facilitate, and even enable, the effective and efficient content-based access and manipulation of multimedia data by providing a standardized description interface.

A decoder that is compliant with both MPEG-4 and MPEG-7 will enable efficient and highly interactive multimedia applications. Consider our example of the visual scene shown in Fig. 1. While watching the TV, the user may want to search for "shirts" that have similar texture to the fish shown on the screen. Because of the object-based representation provided by MPEG-4, the "fish" object can be easily accessed by the user. Also, since MPEG-4 allows the embedding of user data in the bit stream, it is possible to attach MPEG-7 standardized texture descriptors to the corresponding object bit stream. Therefore, the user can access a database without performing expensive decoding,

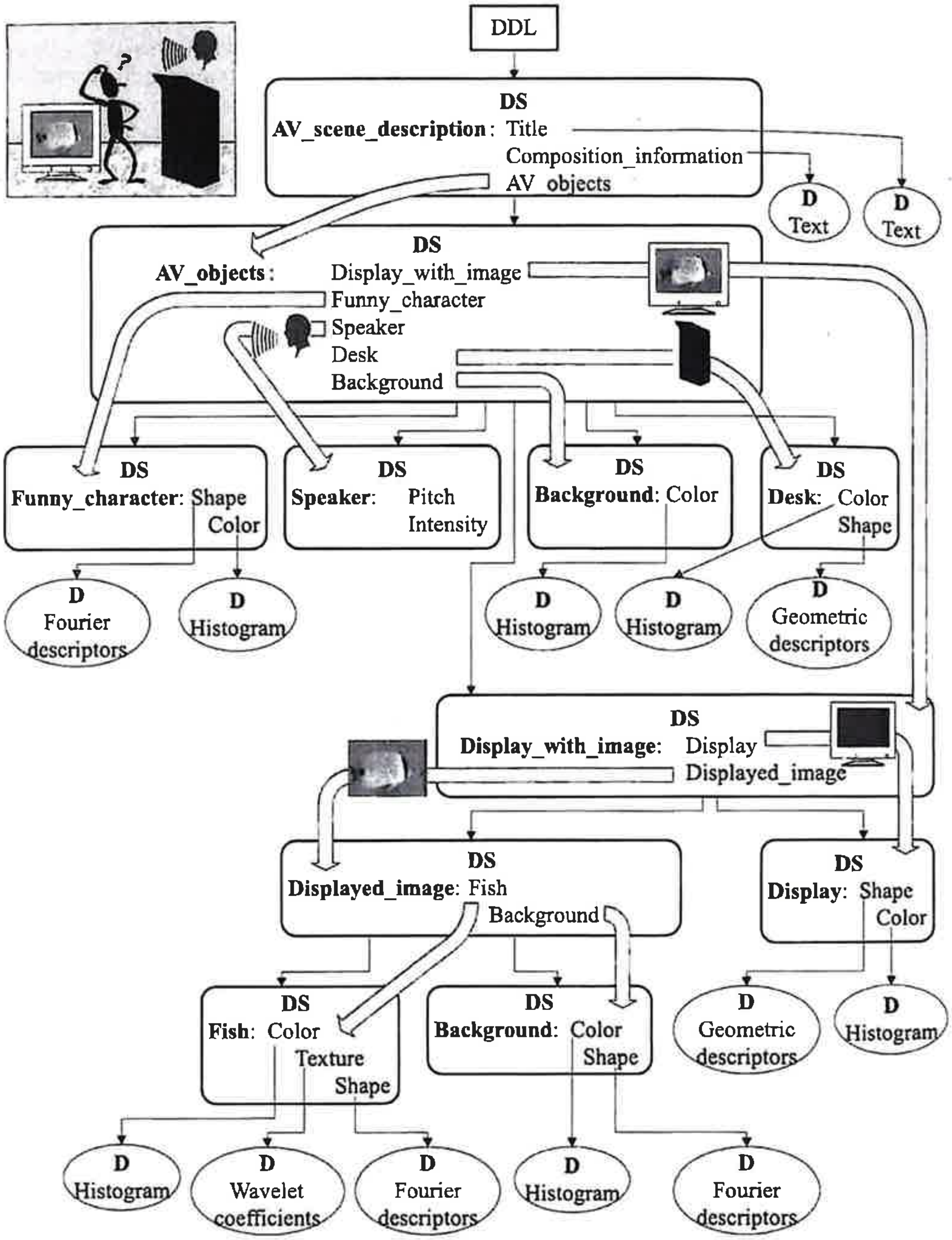


FIGURE 17 Example of description associated with the audiovisual scene.

segmentation, and feature extraction, which would have been required with other representations (e.g., JPEG, MPEG-1/MPEG-2). The user may also want to search for video sequences that contain persons who are "walking". If the underlying bit stream were compliant with MPEG-2, the only way to achieve this would be to decode the bit stream, reconstruct the video sequences, perform spatiotemporal segmentation, and estimate the motion field corresponding to the person video object. On the other hand, the MPEG-4 mesh model can accurately represent continuous motion. Assuming MPEG-7 standardizes mesh motion, the corresponding descriptors can be used by the user to search for objects with similar motion trajectories. Another case is that in which the user wants to search for persons who are "smiling". MPEG-7 may standardize descriptors that are expressed in terms of MPEG-4s FAPs, described in the previous section. Since it is possible to tell the mood of the speaker (e.g., joyful, sad, angry) by the FAPs, the search for a "smiling" person can be easily performed, again without performing expensive processes, such as decoding, segmentation, and feature extraction. These expensive processes have to be performed only once at the encoder end, making MPEG-7/MPEG-4 compliant systems well suited for many applications.

Together, MPEG-4 and MPEG-7 will provide a complete multimedia system solution by allowing the efficient and effective representation, exchange, storage, access, and manipulation of multimedia data. They are expected to enable key technologies for the new generation multimedia applications, revolutionizing our multimedia world.

References

[1] ISO/IEC, "Information technology — coding of moving pictures and associated audio for digital storage media at up to about 1.5 mbits/s: video," 11172–2, 1993.

[2] ISO/IEC, "Information technology — generic coding of moving pictures and associated audio information: video," 13818–2, 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] J. Osterman and A. Puri, "Natural and synthetic video in MPEG–4," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing* (IEEE, New York, 1998), Vol. 5, pp. 3805–3808.

[6] ISO/IEC JTC1/SC29/WG11, "MPEG-4 overview," N2323, July 1998.

[7] MPEG-4 Systems Group, "Coding of audio-visual objects: systems," ISO/IEC JTC1/SC29/WG11 N2201, May, 1998.

[8] MPEG-4 Video Group, "Coding of audio-visual objects: video," ISO/IEC JTC1/SC29/WG11 N2202, March, 1998.

[9] MPEG-4 Audio Group, "Coding of audio-visual objects: audio," ISO/IEC JTC1/SC29/WG11 N2203, March, 1998.

[10] F. Pereira, "MPEG-4: a new challenge for the representation of audiovisual information," in *Picture Coding Symposium '96* (Melbourne, Australia, 1996), pp. 7–16.

[11] T. Sikora, "MPEG-4 very low bit rate video," in *Proceedings of the IEEE International Symposium on Circuits and Systems* (IEEE, New York, 1997).

[12] ISO/IEC JTC1/SC29/WG11, "MPEG-4 systems FAQ, version 7.0a," N2527, October, 1998.

[13] ISO/IEC JTC1/SC29/WG11, "MPEG-4 video FAQ," N1713, April, 1997.

[14] ISO/IEC JTC1/SC29/WG11, "MPEG-4 requirements, version 9," N2456, October, 1998.

[15] ISO/IEC JTC1/SC29/WG11, "Description of MPEG-4," N1410, October, 1996.

[16] T. Sikora, "MPEG-4 video and its potential for future multimedia services," in *Proceedings of the IEEE International Symposium on Circuits and Systems* (IEEE, New York, 1997).

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

[18] MPEG-4 Video Group, "MPEG-4 video verification model version 8.0," ISO/IEC JTC1/SC29/WG11 N1796, July, 1997.

[19] R. M. Haralick and L.H. Shapiro, "Image segmentation techniques," in *Journal of Computer Vision, Graphics and Image Processing* (Academic Press, 1985), pp. 100–132.

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

[21] M. Lee, W. Chen, C. B. Lin, C. Gu, T. Markoc, S. Zabinsky, and R. Szeliski, "A layered video object coding system using sprite and affine motion model," in *IEEE Trans. CSVT* 7, 130–146 (1997).

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

[23] P. Kalra, A. Mangili, T. N. Magnenat, and D. Thalmann, "Simulation of facial muscle actions based on rational free form deformations," in *Proc. Eurographics*, pp. 59–69 (1992).

[24] P. Doenges, T. Capin, F. Lavagetto, J. Ostermann, I. S. Pandzic, and E. Petajan, "MPEG-4: Audio/video and synthetic graphics/audio for real-time, interactive media delivery," *Image Commun.* May, 433–463 (1997).

[25] B. G. Haskell, P. G. Howard, Y. A. Lecun, A. Puri, J. Ostermann, M. R. Civanlar, L. Rabiner, L. Bottou, and P. Haffner, "Image and video coding — emerging standards and beyond," *Proc. CSVT* **8**, pp. 814–837 (1998).

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

[27] R. Talluri, "Error resilient video coding in the MPEG-4 standard," *IEEE Commun. Mag.* **26**, pp. 112–119 (1998).

[28] Microsoft, "MPEG-4 video encoder/decoder," http://drogo.cselt.stet.it/ufv/leonardo/mpeg/public/mpeg-4_fcd/Visual/Natural, 1998.

[29] ISO/IEC JTC1/SC29/WG11, "MPEG-7: context and objectives," N2460, October, 1998.

[30] ISO/IEC JTC1/SC29/WG11, "MPEG-7: requirements document ver. 7," N2461, October, 1998.

[31] Y. Rui, T. S. Huang, and S. Mehrotra, "Content-based image retrieval with relevance feedback in Mars," in *Proceedings of the International Conference on Image Processing* (IEEE, Santa Barbara, California, 1997).

[32] Y. Kageyama and H. Saito, "Image retrieval system capable of learning the user's sensibility using neural networks," in *Proceedings of the International Conference on Neural Networks* (IEEE, Texas, US, 1997), pp. 1543–1567.

[33] University of Pisa, "Icon browser," http://www.cli.di.unipi.it/iconbrowser/, 1998.

[34] Excalibur Technologies Corporation, "Image surfer," http://www.interpix.com, 1998.

[35] Carnegie Mellon University, "Lycos media," http://www.lycos.com/picturethis/, 1998.

[36] Imagiware Inc., "Virtual image archive," http://www.imagiware.com/via/search.html/, 1998.

[37] Excalibur Technologies Corporation, "Yahoo image surfer," http://ipix.yahoo.com/, 1998.

[38] "Whoopie," http://www.whoopie.com, 1998.

[39] K. Messer and J. Kittler, "Using feature selection to aid an iconic search through an image database," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing* (IEEE, New York, 1997), pp. 2605–2608.

[40] D. Androutsos, K. N. Plataniotis, and A. N. Venetsanopoulos, "Efficient image database filtering using color vector techniques," in

Proceedings of Canadian Conference on Electrical and Computer Engineering (Newfoundland, Canada, 1997), pp. 827–830.

[41] K. Liang and C. J. Kuo, "Progressive image indexing and retrieval based on embedded wavelet coding," in *Proceedings of the International Conference on Image Processing*, Santa Barbara, CA, October, 1997.

[42] M. Beatty and B. S. Manjunath, "Dimensionality reduction using multi-dimensional scaling for content-based retrieval," in *Proceedings of the International Conference on Image Processing*, Santa Barbara, CA, October, 1997.

[43] Y. Q. Chen, "Novel techniques for image texture classification," Ph.D. dissertation (Dept. of Electronics and Computer Science, U. of Southampton, UK, 1995).

[44] P. P. Raghu and B. Yegnanarayana, "Segmentation of Gabor-filtered textures using deterministic relaxation," *IEEE Trans. Image Process.* **5**, 1625–1636 (1996).

[45] R. M. Haralick, K. Shanmugam, and I. Dinstein, "Textural features for image classification," *IEEE Trans. Syst. Man Cybernet.* **SMC–3**, 610–621 (1973).

[46] A. K. Jain and K. Karu, "Learning texture discrimination masks," *IEEE Trans. Pattern Anal. Machine Intell.* **18**, 195–205 (1996).

[47] J. P. Eakins, J. M. Boardman, and M. E. Graham, "Similarity retrieval of trademark images," *IEEE Multimed.* April-June, 53–63 (1998).

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

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

[50] C. Y. Yee, K. Tan, T. S. Chua, and B. C. Ooi, "An empirical study of color-spatial retrieval techniques for large image databases," in *Proceedings of IEEE International Conference on Multimedia Computing and Systems* (IEEE, New York, 1998), pp. 218–221.

[51] Columbia University, "WebSeek—a content-based image and video search and catalog tool for the Web," http://www.ctr.columbia.edu/webseek/, 1998.

[52] E. Saber and A. M. Tekalp, "Integration of color, shape, and texture for image annotation and retrieval," in *Proceedings of the International Conference on Image Processing*, Lausanne, Switzerland, September 16–19, 1996.

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

[54] W. Y. Ma and B. S. Manjunath, "NeTra: A toolbox for navigating large image databases," in *Proceedings of the International Conference on Image Processing*, Santa Barbara, CA, October 26–29, 1997.

[55] T. Chen and R. R. Rao, "Audio-visual interaction in multimedia communication," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing* (IEEE, New York, 1997), pp. 179–182.

[56] ISO/IEC JTC1/SC29/WG11, "MPEG-7: proposal package description," N2464, October, 1998.

[57] ISO/IEC JTC1/SC29/WG11, "MPEG-7 Evaluation process document," N2463, October, 1998.