VIDEO COMPRESSION STANDARDS

Video is being used in a growing number of applications, including digital television, digital video disk (DVD), video telephony and teleconferencing, and video communication over the internet. Uncompressed digital video requires very high data rates. The transmission and storage of this data may be impractical, or even impossible, for many applications. Video compression has been crucial for overcoming many technical bottlenecks such as limited bandwidth and storage capacity and for making many of these applications practical (1–4). For example, one of the video formats to be used in high-definition television (HDTV) in the United States is a progressively scanned, 720 × 1280 square pixel, 60 frames/s video signal, with 24-bits/pixel (8-bits for red, green, and blue), which corresponds to a raw data rate of about 1.3 Gbits/sec (Gb/s). Modern digital-communications techniques for terrestrial (over-the-air) television broadcasting provide a transmission capacity of approximately 20 Mb/s in the 6 MHz bandwidth allocated per channel. Video compression techniques achieve compression ratios of about 70 : 1 in order to transmit the video across the bit-rate-limited channel (5).

In recent years, considerable work has been done toward creating international standards for video compression. Standards provide a number of benefits. A primary benefit of video compression standards is that they facilitate interoperability between equipment designed by different manufacturers, thereby lowering risk for both the consumer and the manufacturers. This results in quicker acceptance and widespread use of digital video technology. In addition, these standards are designed for a large variety of applications, and the resulting economies of scale lead to reduced cost and further widespread use.

Much of the standardization work has been performed under the auspices of the International Telecommunications Union (ITU, formerly the International Telegraph and Telephone Consultative Committee, CCITT) and the International Organization for Standardization (ISO). The first standard to gain widespread acceptance was the ITU H.261, which was designed for videoconferencing over the integrated services digital network (ISDN). H.261 was adopted as a standard in 1990. It was designed to operate at $p = 1, 2, \ldots, 30$ multiples of the baseline ISDN data rate, or $p \times 64$ kb/s. At around the same time, the Joint Photographic Experts Group (JPEG) standard for still image compression was finalized by ITU and ISO. The Moving Pictures Expert Group (MPEG) was established by ISO to develop a standard for compressing moving pictures (video) and associated audio on digital storage media such as compact disc-read only memory (CD-ROM). The resulting standard, commonly known as MPEG-1, was finalized in 1991 and achieves approximately VHS (video home system) quality video and audio at about 1.5 Mb/s. A second phase of their work, commonly known as MPEG-2, was an extension of MPEG-1 developed for application toward digital television and for higher bit rates. Currently, the video portion of digital television (TV) and high definition television (HDTV) standards for large portions of North America, Europe, and Asia is based on MPEG-2. A third phase of work, known as MPEG-4, is under development. The goal of MPEG-4 is to provide increased functionality, such as content-based processing and interactivity. Further work has been directed toward the MPEG-7 standard, which also involves video and audio, but rather than considering compression, its goal is a method for describing content to enable efficient multimedia management and searching. In 1993, the ITU initiated a standardization effort with primary goal of videotelephony over the public switched telephone network (PSTN) (conventional analog telephone lines), where the total available data rate is only about 33.6 kb/s. The video compression portion of the standard is H.263 and its first phase was adopted in 1996. An enhanced H.263, known as H.263+, is scheduled to be finalized in 1998, and a long-term version of H.263 is scheduled for standardization in 1999. Table 1 provides a summary of the current and emerging video compression standards.

This article begins by briefly examining some of the system and application issues that directly affect the design of a video compression standard. The principles and practice of conventional video compression algorithms is discussed while concentrating on aspects that are used in the current standards. The conventional video compression standards are described, with emphasis on their primary application profiles and highlighting their differences. Finally, the emerging video compression standards are briefly discussed as well as a number of important problems that arise for compressed video.

SYSTEM ISSUES IN VIDEO COMPRESSION

The design and implementation of a video compression algorithm is directly affected by the system requirements of the particular application. For example, one must consider whether the system will be used for (1) broadcasting or point-

Table 1. Current and Emerging Image and Video Compression Standards

Standard	Application	Bit Rate
JPEG	Continuous-tone still-image compression	Variable
MPEG-1	Video on digital storage media (CD-ROM)	1.5 Mb/s
MPEG-2	Digital television	$>$ 2Mb/s
H.261	Video telephony and teleconferencing over ISDN	$p \times 64$ kb/s
H.263	Video telephony over PSTN	$<$33.6 kb/s
MPEG-4	Content-based processing and communication	Variable

J. Webster (ed.), *Wiley Encyclopedia of Electrical and Electronics Engineering*. Copyright © 1999 John Wiley & Sons, Inc.

to-point communication, (2) real-time or non-real-time communication, and (3) delivery over robust or error-prone environments. In addition, one must consider the bandwidth, computation, and memory resources that will be available throughout the system. Once the system requirements are specified, the details of the coding algorithm can be adjusted for the particular application. For example, tradeoffs can be made in reconstructed video quality, bit rate, complexity, delay, error resilience, and functionality. The possibility of trading off video quality for bit rate is readily apparent; however, it may be advantageous to consider tradeoffs among other parameters as well.

Complexity of encoders and decoders can be measured in computational requirements, memory requirements, chip area, power requirements, or simply cost. The desired relative complexities of the encoder and decoder can vary greatly based on the particular application. For example, in television broadcast applications such as HDTV, there will be few encoders and many decoders. In this environment, the encoders can be complex, but the decoders must be available at lower costs. On the other hand, in applications such as two-way video conferencing, devices must be able to encode and decode video simultaneously. In this environment, the relative encoder and decoder cost is less of an issue.

Live applications such as two-way video conferencing require the delay of the encode/transmit/decode cycle to be quite small (e.g., no longer than a few hundred milliseconds). Alternatively, in video storage applications where video is precompressed and stored for future use, encoding delay is not an issue. In these applications, the video may even be coded in multiple passes to optimize the compression performance.

Another consideration is the error resilience of the coded bit stream. Error resilience is very important for video delivery over packet networks and other error-prone environments such as wireless video. However, error resilience may be less critical in reliable video storage applications.

The new digital television broadcast industry also has a set of requirements imposed by the current analog television broadcast industry. Television viewers have come to expect video cassette recorder (VCR) type functionalities such as fast forward and reverse play. In addition, viewers expect quick random access capabilities when changing channels. Content providers expect ad-insertion capabilities, and television studios expect video-editing capabilities. The details of the video compression algorithm should be designed to support these functionalities.

In this section, we showed a few examples of tradeoffs that can be made when designing a video compression algorithm. In the following sections, we describe general principles of video compression that are used in a number of video compression standards. We then show a high-level architecture of the video encoder and decoder that is common to the MPEG-1, MPEG-2, H.261, and H.263 standards. Even though the high-level architectures of these standards are the same, the details of each standard are different. In essence, each standard provides a set of tradeoffs, some of which were described above, so that it is better suited for its target applications and environments.

PRINCIPLES AND PRACTICE OF VIDEO COMPRESSION

The evolution of image and video compression technologies has resulted in a number of general principles and techniques that are used throughout video compression (1–5). For this reason, many video compression standards are based on similar algorithms and have similar high-level architectures. However, the details of each standard differs based on the specific applications for which it was targeted. This section discusses the aspects of video compression that are common to the MPEG-1, MPEG-2, H.261, and H.263 standards. All these standards are lossy, in the sense that the reconstructed video is not exactly equivalent to the original video. In most applications, the reconstructed video does not need to be identical to the original video, but it is important to minimize the viewer's perceived loss by exploiting properties of the video signal and the human visual system.

Video compression consists of a number of interrelated steps including (1) sampling and digitizing the input video signal, (2) processing the digitized video signal with color-space, temporal, and spatial processing methods, and (3) quantizing and coding the processed signal into a compressed bitstream. This section begins by describing the digitized video signal, the properties of the video signal that the compression algorithms attempt to exploit, namely redundancy and irrelevancy, and a general framework for many video compression systems. Then, the temporal, spatial, and color space processing methods are examined in detail. Finally, the quantization and codeword assignment methods used to achieve the compression and produce the compressed bitstream are described.

Representing the Video Signal

A video signal is a continuous function of time, space, and wavelength. This signal must be discretized for digital processing and transmission. This intrinsically involves sampling and quantizing the video signal along each of these dimensions. Temporal sampling is used to create the individual frames or fields of video. Spatial sampling of these frames or fields results in image samples, which are often referred to as picture elements or *pixels.* The temporal and spatial sampling structures may be independent, or they may be coupled together as in interlaced scanning. To represent color, video is usually modeled as the additive combination of three primary colors; red, green, and blue. Each image sample is composed of three color components, each with finite accuracy (often 8 bits per color sample).

An important issue in representing a video signal is the specific choice of the spatiotemporal sampling structure to be used. Conventional television uses *interlaced* scanning, where the video is split into even and odd fields, composed of even and odd scan lines, respectively. The odd field is acquired, transmitted, and displayed first, and the even field follows. Interlaced scanning was developed in the early days of television, to trade off the limited bandwidth between the spatial and temporal dimensions. Specifically, interlace enabled a doubling of the display rate (important for minimizing the perceived flicker of the display) without reducing the total number of lines per frame. Interlaced scanning results in a number of visual artifacts such as interline flicker complicates video processing and general interoperability with computers. Another approach currently used with computer displays is *progressive* scanning, where consecutive scan lines within each frame are read sequentially. An entire frame of video is sampled at one time, rather than splitting the video signal into its even and odd fields, thereby eliminating the interlace artifacts. In principle, the acquisition, processing, communication, and display of a video signal may be performed independently with an appropriate sampling structure for each. All the current video compression standards support progressive scanning, and MPEG-2, which was designed for digital television, supports both progressive and interlaced scanning.

Redundancy and Irrelevancy

One goal of video compression is to reduce the bit rate needed to represent the video so that it can be transmitted or stored with greater efficiency. The reduction in bit rate is achieved by identifying and exploiting the *redundancy* and *irrelevancy* inherent to the video signal. Sources of redundancy include

- temporal: Most frames are highly correlated with their neighbors.
- spatial: Nearby pixels are often correlated with each other.
- color space: RGB components are correlated among themselves.

Two consecutive frames of a video sequence are shown in Fig. 1. The *temporal redundancy* is evident by observing the similarities between the two frames. Within each frame, the *spatial redundancy* is evident in the large regions that have similar characteristics, such as objects and background areas. This illustrates the redundant information inherent to a typical video sequence. Repeatedly transmitting the same information would be a wasteful use of channel bandwidth. Compression can be achieved by reducing the redundancy in the video signal, thereby resulting in a coded bit stream with a lower data rate.

Another significant attribute of video compression that is not applicable to all source coding applications is the realization of what is perceptually relevant and what is not. Even though redundancies are relatively easy to pinpoint and exploit for compression, identifying what is relevant or irrelevant is much more difficult because the human visual system (HVS) is a complex biological process that does not lend itself easily to analytical modeling. An active area of research is the study of human perception and the associated masking phenomena in order to develop algorithms that more efficiently exploit the HVS. In many contexts, it is useful to view irrelevancy as a form of perceptual redundancy, where an element is represented with more resolution than is perceptually required.

General Video Compression System

A video compression system is composed of three distinct, though interrelated, operations: signal representation, quantization, and codeword assignment. The goal of the first operation is to express the video signal in a representation that better facilitates compression. Temporal, spatial, and color space processing are used to create a representation that concentrates the signal energy into a small number of parameters. After processing, only a small fraction of the data must be transmitted for an accurate reconstruction of the signal. The second operation, quantization, performs the discretization of the representation information, and the third operation assigns to the quantized parameters appropriate codewords for efficient transmission. The first and third operations may be performed in a lossless manner, with any loss of information being localized solely within the quantization operation. By isolating the potential loss of information in a single operation, a much simpler design process and fine-tuning of the system are possible.

The rest of this section examines how these operations are performed within the current video compression standards. This section continues by closely examining the temporal, spatial, and color space processing that is typically performed on a video signal to identify and exploit its redundancy and irrelevany and thereby create an efficient representation of the video. It then discusses the quantization and codeword assignment used in these standards. While reading this section, it may be beneficial to refer frequently to Figs. 2 and 3, which illustrate typical video encoder and decoder architectures. These architectures form the basic framework for all the current video compression standards. As illustrated in Figs. 2 and 3, typical compression algorithms begin with color space processing followed by temporal and then spatial processing. Temporal processing is discussed first, followed by spatial and color space processing.

Temporal Processing

A video sequence is a series of still images shown in rapid succession to give the impression of continuous motion. Although each frame is distinct, the high frame rate necessary to achieve proper motion rendition usually results in significant temporal redundancy among adjacent frames. Temporal processing attempts to exploit this redundancy.

Processing each frame individually without taking into account the temporal dimension of the video (that is, independently of the other frames) is called *intraframe processing.* Processing a frame while exploiting the temporal dimension of the video is called *interframe processing.* Purely intraframe processing provides a number of benefits over interframe processing: (1) lower computational complexity, (2) fewer frame stores requirements at the encoder and the decoder, (3) simpler random access into the compressed bit stream, and (4) better resilience to transmission errors. However, because intraframe encoding does not exploit the temporal redundancies in the video, it cannot achieve the high compression rates achievable with interframe methods. With today's rapidly declining memory and computation costs and limited bandwidth availability, some amount of temporal processing is an essential ingredient of many video compression systems.

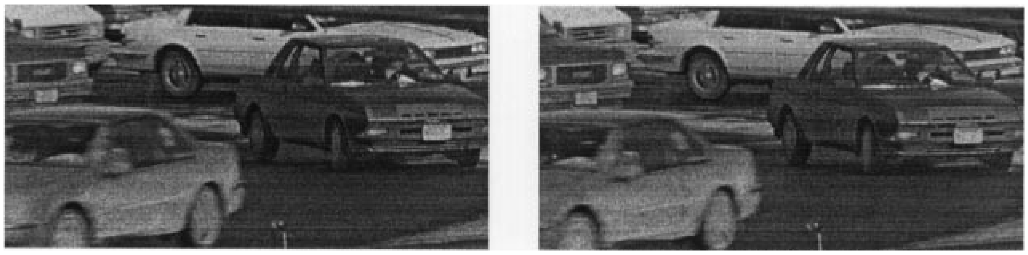

Figure 1. Two consecutive frames of a video sequence. The temporal redundancy is evident by observing the similarities between the frames. The spatial redundancy is evident by observing the large spatial regions in each frame that have similar characteristics.

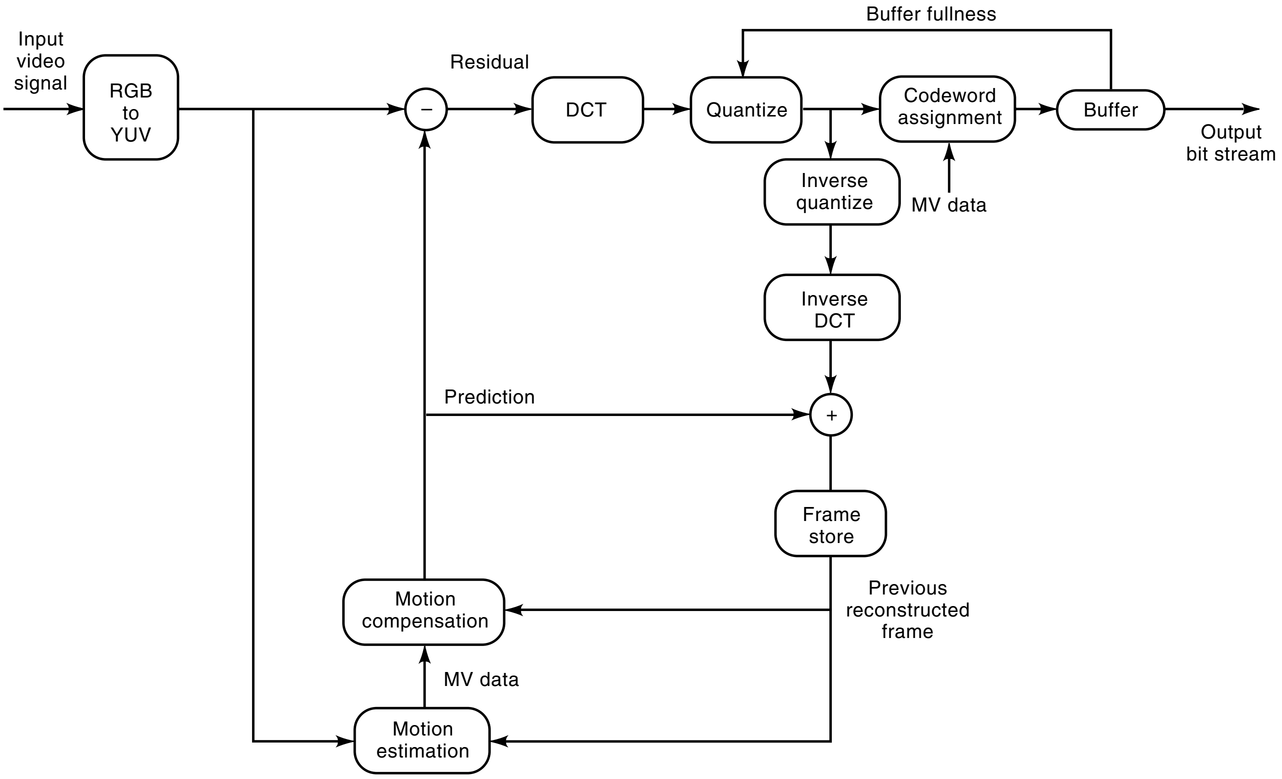

Figure 2. A high-level view of a typical video encoder.

Consecutive video frames typically contain the same imagery, although possibly at different spatial locations. This temporal redundancy (predictability) can be exploited by coding a given frame and then using it to form a prediction for the next frame, while compensating for the motion between the two frames. To accomplish this, an initial frame must be coded independently of the other frames and transmitted to the decoder. Then the motion between the coded frame and the current frame to be coded must be estimated, and an appropriate prediction of the current frame must be made. The error in the prediction, or *residual,* is then coded and transmitted. The process of estimating the motion between frames is known as *motion estimation* (ME). The general processing of individual frames while compensating for the presence of motion is called *motion-compensated* (MC) processing, and forming a prediction while compensating for motion is known as motion-compensated prediction or *MC-prediction.* This section continues by providing a brief overview of conventional motion estimation algorithms and causal and bidirectional motion-compensated prediction, as used in current video compression standards.

Motion Estimation. In motion estimation, the same imagery is assumed to appear in consecutive video frames, although possibly at different spatial locations. The motion may be global, as in a camera pan, or local within the frame, as in a moving object. To optimize ME performance, an estimate of the motion is computed for each local region within a frame. The most common model for the local motion is simple translational motion. This model is highly restrictive and cannot represent the large number of possible motions, such as rotations, scale changes, and other complex motions. Nevertheless, by assuming translational motion only locally and by identifying regions where the model succeeds or fails, excellent coding performance can be achieved.

One approach for performing ME is based on *block-matching* methods. In block matching, the current frame is partitioned into rectangular regions or blocks of pixels, and a

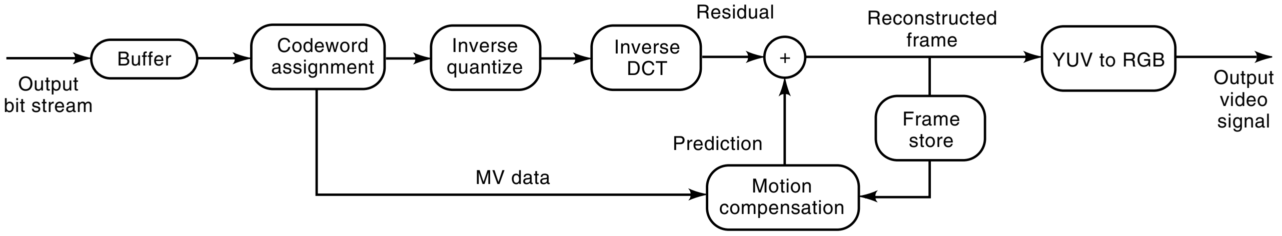

Figure 3. Typical video decoder.

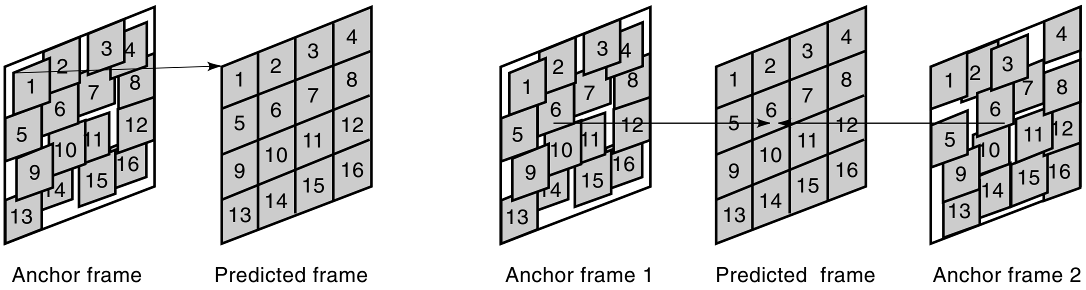

Figure 4. Block-based forward and bidirectional motion-compensated prediction is illustrated on the left and right, respectively. The current frame to be coded is partitioned into blocks. For each block, a prediction is formed by finding the best match in previously coded reference frames.

search is performed to find the displacement that provides the "best match" among possible blocks in a nearby frame, hence the term block matching. The offset or displacement of the best match is represented by a motion vector, which is coded into the bitstream so that it can be used in the decoder. Current video compression standards are nearly universally based on block-matching ME and MC-prediction algorithms. This is because block matching achieves high performance while also exhibiting a simple, periodic structure that simplifies VLSI implementation.

A number of important issues arise when designing a block-matching scheme for video compression. The displacement or *motion vector* for the "best-matching" block can be estimated by maximizing the similarity (e.g., normalized correlation) between blocks, or by minimizing the dissimilarity [e.g., mean square error (MSE) or mean absolute error (MAE)] between blocks. Of the different decision metrics, MAE is often chosen because it achieves similar performance as the others, but without requiring any multiplications. Choosing the size of the block is a tradeoff between the benefits of a higher resolution motion field (improved prediction/interpolation) and the amount of information required to describe it. Similarly, choosing the search range in the reference frame to search for a match trades off improved ability to track fast motion, such as in sporting events, for a greater number of candidate matches that must be examined. Estimating the motion to subpixel accuracy can also enhance performance, but it requires spatial interpolation to determine the noninteger spaced image samples. Conventional video compression standards use 16 × 16-pixel blocks and support motion vectors estimated to half-pixel accuracy.

The motion vector for the best match may be found in a brute force yet straightforward manner by examining every possible candidate within the search area. This method is called *full search* or *exhaustive search,* and it ensures the best match within the reference area. As an alternative to the large computational requirements of the exhaustive search method, adaptive methods that efficiently search for a minimum by evaluating a reduced number of possible displacements may be applied. Hierarchical or multigrid approaches may also be employed to reduce the computational requirements of ME. In these approaches, a low-resolution version of the video is used to produce an initial coarse estimate of the motion. This estimate is subsequently refined using higher-resolution versions of the video.

Motion-Compensated Prediction. The temporal redundancy inherent in a video signal can be exploited with MC-prediction. Through a block-matching ME algorithm, each block of the current frame can be predicted based upon a translation of a block from a reference frame. In causal or forward MC-prediction, the reference or anchor frame is a preceding previously coded frame. For noncausal bidirectional MC-prediction, two anchor frames are used as reference, one preceding and one following the predicted frame.

The process of causal or forward MC-prediction is illustrated in Fig. 4. The frame to be encoded is partitioned into blocks, and a prediction is formed for each block with the best-matching block from the reference or anchor frame. Because the prediction is seldom perfect, the prediction error or residual is further processed using spatial domain techniques. ME and MC-prediction as described previously were applied

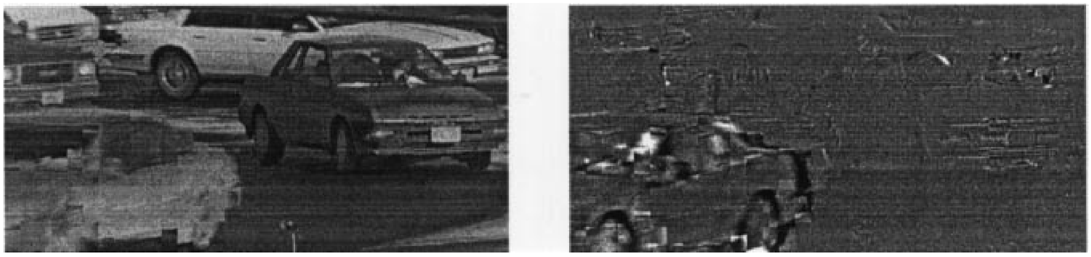

Figure 5. Block-based motion estimation and motion-compensation prediction applied to the two video frames shown in Fig. 1. The resulting prediction and error signals are shown on the left and right, respectively. The amplitude of the error signal has been scaled and offset so that gray corresponds to zero amplitude; white, a large positive amplitude; and black, a large negative amplitude.

to the two video frames shown in Fig. 1. The predicted frame and the resulting prediction error are shown in Fig. 5.

The process of noncausal or bidirectional MC-prediction is shown in Fig. 4. Once again, the frame to be encoded is partitioned into blocks. However, in this case, three predictions (forward, backward, and bidirectional) are formed for each block. ME is used to find the best match for each block from each of the two anchor frames. The forward prediction is the best-matching block from the preceding anchor frame, the backward prediction is the best-matching block from the following anchor frame, and the bidirectional prediction is the average of the two. An advantage of bidirectional prediction is that it can exploit the benefits of both forward and backward predictions. For example, a previous frame cannot predict the appearance of new imagery, whereas a future frame cannot predict disappearing imagery. Once again, because the prediction is seldom perfect, the prediction error is further processed with spatial domain techniques.

The predictive and differential coding aspects of MC-prediction means that it is a form of differential pulse code modulation (DPCM) along the temporal dimension. it is adaptive to the video because the motion vectors guide the prediction process. Typically, MC-prediction is highly effective and is a significant source of coding gain in video compression algorithms.

Conventional video compression standards perform the computationally intensive task of motion estimation only at the encoder, and transmit the motion vectors to the decoder which only has to perform motion-compensation.

Design Issues in MC-Prediction. There are many instances when the temporal processing may fail, either globally or locally. For high-quality video compression, it is very important that the system be able to identify these instances and process them appropriately. For example, at a scene change, causal MC-prediction may produce a prediction error that can be more difficult to code than the original frame. In this case, the MC-prediction should be suppressed, and the original frame should be coded with intraframe coding. Similarly, with bidirectional MC-prediction, there are instances when one of the two references may be inappropriate to use, and the prediction should be made using the other reference frame.

Similar issues exist at a local level. For example, the appearance of new imagery in a region may cause the forward-predicted residual to be more difficult to code than the region itself. This may also occur in areas that have motion that is not modeled well by the block MC-prediction process. In these cases, it is important that the system identifies each local region where MC-prediction fails and subsequently performs the appropriate coding. This illustrates the importance of locally adaptive processing, or spatially adaptive inter/intraframe processing.

The choice of using bidirectional MC-prediction also requires careful consideration. Extra frame memory is required for the bidirectional processing of MC-prediction as compared to the forward-only processing of causal MC-prediction. This incurs additional delay in the encoding process because one must "wait" for a later frame to come along before the current frame can be coded. If the difficulties described here are acceptable for the particular application, bidirectional MC-prediction can provide a number of advantages. In the case of moving objects in natural scenery, a better prediction of the current frame can be formed by considering previous and later frames. For example, an object that becomes uncovered or visible in a current frame cannot be predicted with an earlier frame, but it can be predicted from a later frame.

Another advantage of bidirectional MC-prediction is that it provides a form of temporal scalability. Consider a video sequence where every other frame is coded as a B frame. The disposable nature of B frames (no other frames depend on a B frame) means that by simply discarding the B frame data, one can reconstruct the video sequence with half the frame rate.

The recursive nature of predictive coding schemes means that it is essential that the decoder accurately track the encoder. If they become unsynchronized, the prediction at the decoder will not match the prediction at the encoder, and the whole process will fail. For example, in digital television, this issue arises when considering receiver initialization and channel acquisition (when the receiver is turned on or the channel is changed), and when uncorrectable channel errors occur. With the DPCM-style MC-prediction, an initial frame must be available at the decoder to (re)start the prediction loop. Therefore, a mechanism must be built into the system so that if the decoder loses synchronization for any reason, it can rapidly reacquire tracking. One popular solution is periodic intracoding of an entire frame, thereby producing a periodic reinitialization of the temporal prediction at both the encoder and the decoder.

Spatial Processing

Applying MC-processing reduces the temporal redundancy of the video signal, but spatial redundancy still exists within the MC-residual. This is especially true if no MC-processing is performed and the original frame itself is to be coded. There exist a variety of methods for reducing the spatial redundancy in an original frame or a MC-residual, where the most popular are transform and subband filtering schemes.

Transform schemes are based on the idea that an image can be linearly transformed into another domain where most of the energy (and information) is concentrated in a small fraction of the transform coefficients. Coding and transmission of these few energetic coefficients may then result in a high-quality reconstruction. Subband filtering schemes process an image by filtering it into separate frequency bands or subbands, and each subband can be adaptively encoded in order to exploit its specific characteristics. Popular subband filtering schemes include the lapped orthogonal transform and the wavelet transform. As suggested by these names, there exists a very close relationship between transform and subband filtering. Of all the transform and subband filtering schemes, the 8×8 Block Discrete Cosine Transform is nearly universally used in current video compression standards.

Discrete Cosine Transform. The Discrete Cosine Transform (DCT) is very similar to the Discrete Fourier Transform (DFT), but it eliminates the artificial discontinuity inherent in computing the DFT and thereby yields improved energy compaction for typical images. The very good energy compaction and decorrelation properties of the DCT coupled with its fast computational implementation have resulted in its extensive study and use in image and video compression standards. Even though the DCT may be computed for the entire frame, much improved performance can be achieved by partitioning the frame into numerous smaller regions, each of which is independently transformed and adaptively processed. For example, computing the DCT of the entire frame results in the whole frame being processed together. However, in typical video, the characteristics vary considerably over the spatial extent of each frame and from frame to frame. In order to exploit the varying spatial characteristics of the video signal, each frame is typically partitioned into 8 × 8 blocks, which are independently transformed and *adaptively processed* to exploit their individual characteristics. The application of the DCT in this manner is often referred to as the Block DCT. Spatially adaptive processing, which the Block-DCT facilitates, is one of the most important ingredients for a high-performance video compression system. Partitioning a frame into small blocks before computing the transform also affords other benefits, including reduced computational and memory requirements. In addition, the two-dimensional DCT of an 8 × 8 block can be efficiently computed by applying the one-dimensional DCT separably to the rows and the columns of the block.

The DCT coefficients express the block of pixels as a linear combination of spatial frequencies. The "DC" or (0,0) coefficient expresses the average value of the block, while the other coefficients express the higher horizontal and vertical spatial frequencies in the block. Interpretating the DCT coefficients in terms of spatial frequencies is very important since the human visual system is generally less sensitive to errors in the high frequencies than low frequencies, and therefore the high frequencies can be quantized more coarsely (with lower precision) than the low frequencies.

The Block DCT is used nearly universally in current video compression standards because it provides excellent performance with low complexity. This is partially because the DCT provides very good energy compaction for typical images. In addition, its block-based structure enables (1) simple spatially adaptive processing on a block by block basis including simple inter/intra processing, (2) convenient interfaces with block-based MC-prediction (the MC block boundaries line up with the DCT block boundaries), and (3) simple implementations with low computation and memory requirements. Furthermore, because of the widespread use of the Block DCT in image and video compression standards, there are significant benefits to its continued use, such as simplified hardware and software interoperability among standards. The DCT may be replaced by another transform in the future (e.g., wavelet transform); however, currently it is an important element in video compression standards.

Color Space Processing

A video compression system should account for the color nature of a video signal. A video compression algorithm may approach the problem as compressing a monochrome video signal and simply apply the same processing steps to each of the three color components that comprise a color video signal. However, this would be very inefficient because the three color components, Red, Green, and Blue (RGB), are highly correlated with each other. More importantly, the human visual perception differs for the luminance (intensity) and chrominance characteristics of a video signal. To reduce the correlation among the RGB components and to enable the compression system to exploit the differing perceptual sensitivity to the luminance and chrominance characteristics, a *color space conversion* is usually performed.

The goal is to convert the RGB color space to a domain where the differences in the HVS response can be exploited. Typically, this is accomplished through a linear transformation to the YIQ (NTSC) or YUV (SMPTE 240M colorimetry standard) color spaces. Y corresponds to the luminance (intensity or black and white picture), whereas I and Q or U and V correspond to the chrominance. The HVS has reduced perceptual sensitivity to the chrominance components and, with this representation, it can be easily exploited in the quantization operation. Similarly, the HVS has reduced spatial frequency response to the chrominance as compared to the luminance components. This characteristic can be exploited through a reduced sampling density for the chrominance components. For example, the chrominance may be decimated by a factor of 2 along both the horizontal and vertical dimensions, producing components that are one-quarter the spatial resolution of the luminance. However, for high-performance compression system, retaining the full chrominance resolution does not require much capacity and may be beneficial for some applications such as computer-generated graphics or encoding text containing saturated colors.

When performing ME on color video, the motion field may be computed from all three components. Alternatively, the motion field may be computed only from the luninance component and applied to both the luminance and chrominance components. This procedure eliminates the computationally expensive task of estimating the motion for each chrominance component by exploiting the significant correlation between movement among the different color planes. This method may fail in an isoluminance situation, when adjacent objects of similar luminance but differing chrominance move in different directions. Nevertheless, in general this algorithm performs extremely well.

By applying very simple processing, the differing human visual perception to, and the correlation among, the different color components can be exploited. A significant result is that a three-component color video signal can be coded with less than a 50% increase in capacity over that required for a single-component (monochrome) video signal. Also, subjective tests have shown that more perceptually appealing video may be produced by coding a color video signal at a given rate than by coding a monochrome signal at the same rate.

Quantization

Quantization is used to discretize the various parameter values (e.g., the DCT coefficient amplitudes). Bit rate compression is achieved through quantization and codeword assignment. The quantization process can be the only lossy step in the compression algorithm. This is very important because it simplifies the design process and facilitates fine-tuning of the system.

Quantization may be applied to elements individually (scalar quantization) or to a group or vector of elements simultaneously (vector quantization). Scalar quantization is nearly universally used in current video compression standards. In scalar quantization, each element is quantized with a *uniform* (linear) or *nonuniform* (nonlinear) quantizer. The quantizer may also include a *dead zone* (enlarged interval around zero) to set small, noiselike perturbations of the element value to zero. The close relationship between quantization and codeword assignment suggests that joint optimization is necessary for optimum performance, but this is a highly complex process. However, experiments have shown that a linear quantizer with an appropriate stepsize individually chosen for each element to be quantized, followed by proper entropy coding, may yield close to optimum performance. This will be discussed in the context of quantizing the DCT coefficients.

When quantizing the DCT coefficients, the differing perceptual importance of the various coefficients can be exploited by "allocating the bits" to shape the quantization noise into the perceptually less important areas. This can be accomplished by varying the relative stepsizes of the quantizers for the different coefficients. The perceptually important coefficients may be quantized with a finer stepsize than the others. For example, low spatial frequency coefficients may be quantized finely, whereas the less important high-frequency coefficients may be quantized more coarsely. Similarly, luminance, which is the most visually important component, may be quantized more finely than chrominance. A simple method to achieve different stepsizes is to normalize or weight each coefficient based on its visual importance. All the normalized coefficients may then be quantized in the same manner, such as rounding to the nearest integer (uniform quantization). Normalization or weighting effectively scales the quantizer from one coefficient to another.

In typical signal compression applications, only a few variables are usually quantized to zero. However, in video compression, most of the transform coefficients are quantized to zero. There may be a few nonzero low-frequency coefficients and a sparse scattering of nonzero high-frequency coefficients, but the great majority of coefficients are typically quantized to zero. To exploit this phenomenon, the two-dimensional array of transform coefficients may be reformatted and prioritized into a one-dimensional sequence through a zigzag scanning. This results in most of the important nonzero coefficients (in terms of energy and visual perception) being grouped together early in the sequence. They will be followed by long runs of coefficients that are quantized to zero. These zero-valued coefficients can be efficiently represented through run-length encoding. In run-length encoding, the number of consecutive zero coefficients (runs) before a nonzero coefficient is encoded, followed by the nonzero coefficient value. The run length and the coefficient value can be entropy coded, either separately or jointly. The scanning separates most of the zero and the nonzero coefficients into groups, thereby enhancing the efficiency of the run-length encoding process. In addition, a special End Of Block (EOB) marker is used to signify when all the remaining coefficients in the sequence are equal to zero. This approach is extremely efficient, yielding a significant degree of compression.

Codeword Assignment

Quantization creates an efficient discrete representation for the data. Codeword assignment takes the quantized values and produces a digital bit stream for transmission or storage. The quantized values can be simply represented using *uniform* or *fixed-length codewords,* where every quantized value will be represented with the same number of bits. Greater efficiency, in terms of bit rate, can be achieved by employing *entropy coding.* Entropy coding attempts to exploit the statistical properties of the signal to be encoded. A signal, whether it is a pixel value or a transform coefficient, has a certain amount of information, or entropy, based on the probability of the different possible values or events occurring. For example, an event that occurs infrequently conveys much more new information than one that occurs often. By realizing that some events occur more frequently than others, the average bit rate may be reduced.

A number of important issues arise in regard to the use of entropy coding. Entropy coding coupled with the nonstationarity of the video signal results in a time-varying bit rate. (Other aspects of the source coding may also lead to a variable bit rate, including the use of run-length coding and end-of-block marker.) Therefore, a buffer and a buffer control mechanism are necessary if the variable bit rate source coder is to be coupled with a constant bit rate channel. In addition, entropy coding makes it more difficult to recover from bit errors or lost packets in an error-prone environment. Nevertheless, the sizeable decrease in bit rate that may be achieved with entropy coding has lead to its widespread use in image and video compression standards. Specifically, these compression algorithms typically employ a judicious choice of fixed-length coding and variable-length coding (entropy coding) for the variety of elements that must be coded.

Entropy Coding. An entropy coder is used to reduce the statistical redundancy inherent in the parameters encoded for transmission. The primary redundancy is the nonuniform probability distribution over the possible range of each parameter. The more the probability distribution deviates from a uniform distribution, the greater improvement can be achieved via entropy coding. Other sources of statistical redundancy that may exist include the statistical dependence among the encoded parameters.

Currently, the most popular entropy coding approaches are *variable length coding* schemes such as *Huffman coding.* In Huffman coding, a codebook that minimizes the entropy subject to the codeword constraints of integer lengths and unique decodability is generated. Events that are more likely to occur will be assigned shorter length codewords than those that are less likely to occur. Huffman coding provides a reduction in the average bit rate. Good performance is achieved by using a few codebooks where parameters with similar statistics are grouped and encoded together. Similarly, the size of each codebook can be reduced by grouping together very unlikely events into a single entry within the codebook. When an event in this group occurs, the codeword for this group is transmitted followed by an exact description of the event.

VBR, CBR, and Buffer Control

Whenever entropy coding is employed, the video encoder will produce a variable bit rate (VBR) output based on the video statistics. If the application requires a constant bit rate (CBR) output, a buffer and buffer control mechanism is necessary to couple the two. The buffering must be carefully designed. Random spikes in the bit rate can overflow the buffer, whereas dips in the bit rate can produce an underflow. What is needed is some form of buffer control that would allow efficient allocation of bits to encode the video while ensuring that no overflow or underflow occurs.

The buffer control typically involves a feedback mechanism to the compression algorithm whereby the amplitude resolution (quantization) and/or spatial, temporal, and color resolution may be varied in accordance with the instantaneous bit rate requirements. The goal is to keep the average bit rate constant and equal to the available channel rate. If the bit rate increases significantly, the quantization can be made coarser to reduce it. If the bit rate decreases significantly, a finer quantization can be performed to increase it. When discussing the average bit rate, it may be considered over the entire frame (global buffer control) or over a local region (local buffer control). Global buffer control has the advantage of appropriately allocating the bit rate over the entire frame, resulting in the highest performance and ensuring uniform video quality over the entire frame. With local buffer control, it is more difficult to achieve these results, but it may yield a simpler solution. In addition, in some cases such as storage, it is possible to first analyze the entire video sequence and decide how to distribute the bits along the entire video sequence before actually performing the compression. These multi-pass algorithms (first pass analysis, second pass compression) lead to more effective bit allocation and therefore significantly improved video quality. Bit allocation and buffer control are key elements of any high-performance video compression system and should be an integral part of the design of any such system.

CURRENT VIDEO COMPRESSION STANDARDS

This section presents an overview of the most popular video compression standards currently in use. Specifically, the MPEG-1, MPEG-2, H.261, and H.263 video compression standards are discussed. Each of these standards is based on the compression techniques described in the previous section. This section begins by describing the baseline video encoder and decoder architectures that form the basis for all of these video compression standards. The details of each standard differ based on the target applications for which it was designed, and this section continues by describing the different standards and their application profiles.

Video Encoder and Decoder Architectures

The MPEG-1, MPEG-2, H.261, and H.263 video compression standards are based on motion-compensated prediction and transform coding. High-level views of a typical video encoder and decoder are shown in Figs. 2 and 3. As is discussed in more detail later, the various standards specify the bit stream syntax and the decoding processing, but not the encoder or how the bit stream is actually generated. Therefore, these figures should be viewed only as examples of typical encoders and decoders in a video compression system. In the encoder, the input RGB video signal is transformed to a luminance/chrominance color space (e.g., YUV) to exploit the color space redundancy. To exploit the temporal redundancy, motion estimation and motion-compensated prediction are used to form a prediction of the current frame from the previously encoded frames. The prediction error, or MC-residual, is processed with an adaptive transform encoder. The MC-residual is partitioned into 8 × 8 blocks, and the two-dimensional DCT is computed for each block. The DCT coefficients are quantized in an adaptive manner to exploit the local video characteristics and human perception and to meet any bit rate targets. The quantized coefficients and other information are Huffman coded for increased efficiency. The encoder duplicates the decoder processing to ensure tracking between the two. If the output is to be sent over a CBR channel, then a first-in first-out (FIFO) buffer is used to couple the VBR output of the video encoder to the CBR channel. This is accomplished via a buffer control mechanism whereby the fullness of the FIFO regulates the coarseness/fineness of the coefficient quantization and thereby the video bit rate.

The video decoding process at the receiver is the inverse of the encoding process. (This is shown in Fig. 7.) The bitstream is parsed and Huffman decoded. The nonzero DCT coefficients are identified and inverse quantized. An inverse block DCT operation produces the residual signal, which is combined in a spatially adaptive manner with the previously reconstructed frame to reconstruct the current frame. Finally, the reconstructed frame is converted back to the RGB color space to produce the output video signal.

MPEG-1 and MPEG-2

The Moving Pictures Expert Group (MPEG) was originally established by ISO to develop a standard for compression of moving pictures (video) and associated audio on digital storage media (e.g., CD-ROM). The resulting standard, commonly known as MPEG-1, was finalized in 1991 and achieves approximately VHS quality video and audio at about 1.5 Mb/s (6–8). A second phase of their work, commonly known as MPEG-2, was originally intended as an extension of MPEG-1 developed for application toward interlaced video from conventional television and for bit rates up to 10 Mb/s (8,9). A third phase was envisioned for higher bit rate applications such as HDTV, but it was recognized that those applications could also be addressed within the context of MPEG-2; hence, the third phase was wrapped back into MPEG-2 (consequently, there is no MPEG-3 standard). Both MPEG-1 and MPEG-2 are actually composed of a number of parts including video, audio, systems, and compliance testing. The video compression parts of these standards are often referred to as MPEG-1 video and MPEG-2 video, or MPEG-1 and MPEG-2 for brevity. Currently, MPEG-2 video is on the brink of wide acceptance because it has been adopted as the video portion of the digital TV and HDTV standards for large portions of North America, Europe, and Asia. MPEG-2 video is also the basis for the DVD standard that has recently been introduced. Currently, there are two other standardization efforts, known as MPEG-4 and MPEG-7, that are being developed, and these are discussed in the section on emerging standards.

The MPEG standards are *generic* in the sense that they are not designed for a specific application. Instead, they specify a set of tools that may be useful for a wider range of applications and the end user can decide which tools are most appropriate for the desired application. An important aspect of MPEG is that it specifies the bit stream syntax and the decoding process, but it does *not* specify the encoding process. Hence, there exists considerable freedom in designing an encoder—the sophistication of the encoder is one of the prime differentiating factors among MPEG implementations. Furthermore, improvements in various aspects of the encoding process, such as improved motion estimation or bit allocation, can be immediately incorporated into the applications as long as the coded bit stream remains standard-compliant.

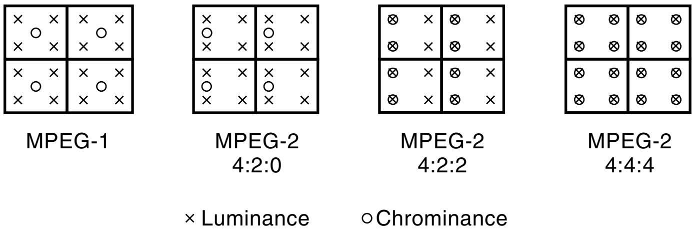

Figure 6. MPEG supports a number of luminance/chrominance formats with different chrominance subsampling patterns.

MPEG-2 is a superset of MPEG-1, supporting higher bit rates, higher resolutions, and interlaced pictures (for television). For interlaced video, the even and odd fields may be coded separately as fields, or a pair of even and odd fields can be combined and coded as a frame. For field-based coding, MPEG-2 provides field-based methods for MC-prediction, Block-DCT, and alternate zigzag scanning. In addition, MPEG-2 provides a number of enhancements including scalable extensions, and tools for improving error resilience and facilitating error concealment. Details on these aspects are not discussed further in this article; the reader is referred to Refs. 8 and 9 for more details on these topics. The following discussion focuses on the salient features of MPEG-1 and MPEG-2 video compression systems, and progressively scanned video is assumed unless mentioned otherwise.

Luminance and Chrominance Sampling. To exploit the human visual system's differing sensitive to luminance and chrominance information, after converting the video signal to a luminance/chrominance color space, the chrominance components may be spatially lowpass filtered and subsampled. MPEG-1 assumes that the horizontal and vertical sampling rates of the chrominance components are half those of the luminance component. The chrominance samples are located in the center of the 2 × 2 blocks of luminance pixels as shown in Fig. 6. MPEG-2 allows other samplings of the chrominance planes. In the MPEG-2 4 : 2 : 0 profile, the chrominance components are also subsampled by factors of two in the horizontal and vertical dimensions. However, unlike MPEG-1, the horizontal offset of the chrominance components are aligned with the luminance components as shown in Fig. 6. The MPEG-2 4 : 2 : 2 profile only subsamples the chrominance component in the horizontal direction. The MPEG-2 4 : 4 : 4 profile assumes that there is no subsampling of the chrominance, and that the chrominance samples are colocated with the luminance samples.

MPEG Coding Structure. MPEG codes video in a hierarchy of units called sequences, groups of pictures (GOPs), pictures, slices, macroblocks, and DCT blocks. These coding units are shown in Fig. 7. MC-prediction is performed on 16 × 16-pixel blocks. A 16 × 16-pixel block is called a macroblock and is coded using 8 × 8-pixel block DCTs and possibly a forward and/or backward motion vector. The macroblocks are scanned in a left-to-right, top-to-bottom fashion, and series of these macroblocks form a slice. All the slices in a frame are comprised of a picture, contiguous pictures form a GOP, and all the GOPs form the entire sequence. The following sections describe each coding unit in greater detail.

MPEG GOPs and Pictures. MPEG uses motion-compensated prediction to exploit the temporal redundancies that may exist in nearby video frames. Video frames are grouped into coding units called groups of pictures. GOPs have the property that they reinitialize the temporal prediction used during encoding. Specifically, the first frame of a GOP is always coded in intraframe mode (independently of other frames) and is called an *I* frame. The remaining frames in the GOP may be coded with intraframe or interframe techniques. The frames coded in interframe mode are predicted with forward or bidirectional prediction and are called P and B frames, respectively. The MPEG syntax allows a GOP to contain any number of frames, but GOP lengths typically range from 9 to 15 frames. A common coding structure for a GOP is shown in Fig. 8.

In this example, the GOP contains nine video frames, I_0 through B_8, where the subscript indicates the frame number. I_9 is the first frame of the next GOP. The arrows indicate the prediction dependencies—the frame at the base of each arrow, the anchor frame, is used to predict the frame at the tip of the arrow, the predicted frame.

I frames are coded independently of other frames. *P* frames depend on a prediction based on the preceding *I* or *P* frame.

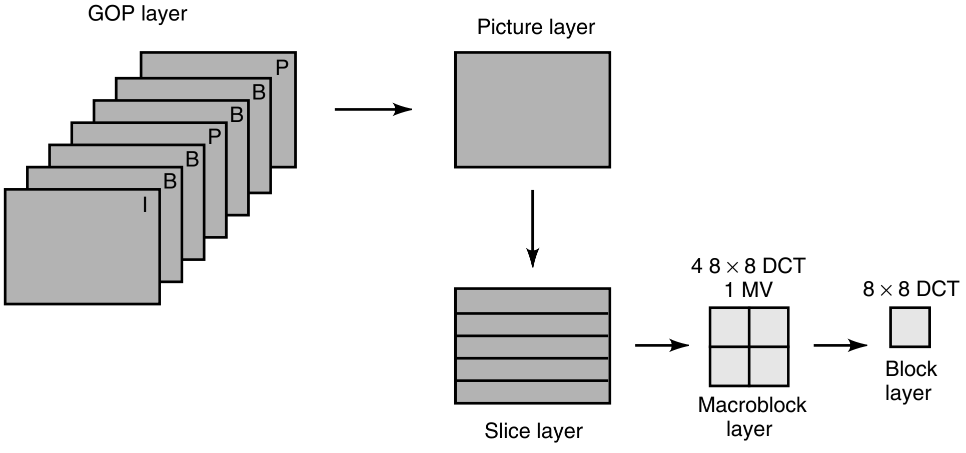

Figure 7. MPEG codes video in a hierarchy of layers. The sequence layer is not shown.

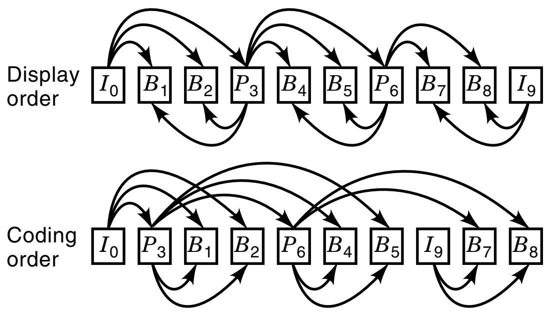

Figure 8. The display and coding orders of the frames in a typical MPEG GOP. The subscripts represent the frame number, and the arrows indicate the prediction dependencies of the coded frames. The coded frames are placed in the bitstream in the coding order, which is the order that the frames are coded and decoded.

B frames depend on a prediction based on the preceding and following *I* or *P* frames. Notice that each *B* frame depends on data from a future frame, which means that the future frame must be (de)coded before the current *B* frame can be (de)-coded. When decoding the bit stream, it does not make sense to receive the data of the coded *B* frame before the data of its two reference frames. For this reason, the coded video data is placed in the data stream in coding order, rather than display order. The *coding order* of a GOP is also shown in Fig. 8.

The GOP structure is flexible and does not have to be fixed. The GOP header does not specify the number of *I*, *P*, or *B* frames in the GOP, nor does it specify its structure—these are completely determined by the order of the data in the stream, which must be compliant with the temporal reference parameters in the bit stream. Thus, there are no rules that restrict the size and structure of the GOP. Of course, care should be taken to ensure that the MPEG syntactic requirements and buffer constraints are satisfied.

The coding of *I*, *P*, and *B* frames typically require different amounts of data. *I* frames require larger amounts of data because they are coded independently of the other frames. *P* and *B* frames typically require less data than *I* frames because of the temporal prediction. *B* frames are often coded with less data than *P* frames for two primary reasons. First, a better prediction can be formed when using both preceding and following reference frames. Second, coding *B* frames at slightly lower quality does not have a negative effect on the quality of other frames. Specifically, because *B* frames are not used in predicting other frames, lower-quality coding of *B* frames will not effect other frames in the sequence. *I* and *P* frames, however, are used as anchor frames when predicting other *P* and *B* frames. Therefore, lower-quality coding of these frames will result in poorer predictions of other frames, thus reducing the overall coding efficiency of the sequence.

MPEG Macroblocks. MPEG uses 16 × 16-pixel MC-prediction to reduce the temporal redundancies inherent in the video. The motion vectors are estimated to half-pixel accuracy, and the MC-prediction at a half-pixel location is determined by bilinear interpolation. The processing is adaptive on a macroblock by macroblock basis, that is, for each macroblock a decision is made as to what is the most appropriate method to process it. As previously discussed, each frames of a video sequence can be coded as an *I*, *P*, or *B* frame. In I frames, every macroblock must be coded in intraframe mode (i.e., prediction is not used). In *P* frames, each macroblock can be coded with either forward prediction or intraframe mode. In *B* frames, each macroblock can be coded with forward, backward, or bidirectional prediction or in intraframe mode. One MV is specified for each forward- and backward-predicted macroblock, whereas two MVs are specified for each bidirectionally predicted macroblock. Thus, each P frame has a forward motion vector field and one anchor frame, whereas each B frame has a forward and backward motion vector field and two anchor frames. Whenever prediction is used, the appropriate motion vectors and the resulting residual are coded into the data stream.

A header at the beginning of the macroblock identifies how it is coded. For example, because some blocks in an intercoded macroblock may be all zero (there are no nonzero quantized coefficients to be transmitted), a coded block pattern can be used to indicate which 8 × 8 blocks contain nonzero coefficients and which blocks contain all zeros.

MPEG DCT Blocks. Each macroblock (intra or inter) is partitioned into 8 × 8 pixel blocks, and the two-dimensional DCT is computed for each block. The DCT coefficients are individually quantized with step sizes appropriately chosen to exploit their differing perceptual importance as well as to exploit the local scene complexity and bit rate targets. The quantized coefficients are then zigzag scanned and run-length coded, and the resulting (run-length, amplitude) pairs are Huffman coded and placed in the bit stream. There is also some interblock and inter-macroblock processing that is performed within each slice, and this is discussed next.

MPEG Slices. The MPEG slice is a coding layer that falls between the picture and macroblock layers. The macroblocks of a picture are scanned in a left-to-right, top-to-bottom order, and groups of contiguous macroblocks form slices. The MPEG profiles require every macroblock to belong to a slice so that all the slices comprise the entire picture. In MPEG-1, slices can begin at any macroblock and can extend over multiple macroblock rows. In MPEG-2, a slice must start at the beginning of each row, and each row can contain multiple slices.

Slices provide a number of advantages. First, they provide a structure for predicting some parameters across macroblocks (thereby resulting in improved compression) while maintaining a level of error resilience. For example, in *I* frames, the DC value of the DCT coefficients may be correlated from block to block. This correlation is exploited by coding the DC coefficient of the first DCT block in the slice as is and coding the DC coefficients of the remaining blocks differentially with respect to the previous DC value. Similarly, in *P* and *B* frames, the motion vectors are coded differentially within a slice. The prediction of the DC coefficients and motion vectors are reinitialized at each new slice, thus maintaining a level of error resilience. If an error occurs in the bit stream, the remaining data in the slice is lost. However, the decoder can recover by searching for the next start code, resynchronizing the bit stream, and continuing the decoding process. Another advantage that slices provide is a convenient structure for adapting the coding parameters to the local characteristics of the video. The slice level provides a good tradeoff between the gain that can be achieved with spatially adaptive processing and the overhead needed to describe the adaptivity.

MPEG Syntax. The syntax of the MPEG-1 data stream has the following structure:

- A sequence header consists of a *sequence start code* followed by *sequence parameters.* Sequences contain a number of GOPs.

- Each GOP header consists of a *GOP start code* followed by *GOP parameters.* GOPs contain a number of pictures.
  - Each picture header consists of a *picture start code* followed by *picture parameters.* Pictures contain a number of slices.
    - Each slice header consists of a *slice start code* followed by *slice parameters.*
      - The slice header is followed by slice data, which contains the coded macroblocks.

The *sequence header* specifies the picture height, picture width, and sample aspect ratio. In addition, it sets the frame rate, bit rate, and buffer size for the sequence. If the default quantizers are not used, then the quantizer matrices are also included in the sequence header. The *GOP header* specifies the time code and indicates whether the GOP is open or closed. A GOP is open or closed depending on whether or not the temporal prediction of its frames requires data from other GOPs. The *picture header* specifies the temporal reference parameter, the picture type (I, P, or B), and the buffer fullness (via the vbv_delay parameter). If temporal prediction is used, it also describes the motion vector precision (full or half pixel) and the motion vector range. The *slice header* specifies the macroblock row in which slice starts and the initial quantizer scale factor for the DCT coefficients. The *macroblock header* specifies the relative position of the macroblock in relation to the previously coded macroblock. It contains a flag to indicate whether intra- or inter-frame coding is used. If inter-frame coding is used, it contains the coded motion vectors, which may be differentially coded with respect to previous motion vectors. The quantizer scale factor may be adjusted at the macroblock level. One bit is used to specify whether the factor is adjusted. If it is, the new scale factor is specified. The macroblock header also specifies a coded block pattern for the macroblock. This describes which of the luminance and chrominance DCT blocks are coded. Finally, the DCT coefficients of the coded blocks are coded into the bit stream. The DC coefficient is coded first, followed by the run lengths and amplitudes of the remaining nonzero coefficients. If it is an intra-macroblock, the DC coefficient is coded differentially.

The sequence, GOP, picture, and slice headers begin with start codes. Start codes are useful because they can be found by simply examining the bit stream; this facilitates efficient random access into the compressed bit stream. For example, one could find the coded data that corresponds to the 2nd slice of the 2nd picture of the 22nd GOP by simply examining the coded data stream without parsing and decoding the data. Of course, reconstructing the actual pixels of that slice may require parsing and decoding additional portions of the data stream because of the prediction used in conventional video-coding algorithms. However, computational benefits can still be achieved by locating the beginning of the 22nd GOP and parsing and decoding the data from that point on thus exploiting the temporal refresh property inherent to GOPs.

MPEG Picture Quality and Rate Control. In MPEG, pictures are coded into data segments with different lengths. However, the frame rate of the displayed sequence is constant. Thus, achieving CBR transmission of the MPEG stream requires using buffers at both the encoder and the decoder. MPEG defines an idealized model of the decoder, referred to as the video buffer verifier, to ensure that the decoder buffer does not overflow or underflow. In CBR transmission, the decoder buffer is filled at a constant rate, and the data for each I, P, or B picture is emptied at regular time intervals corresponding to the frame rate of the sequence. If a picture contains a large amount of data, the buffer empties by a large amount; whereas if a picture contains a small amount of data, the buffer empties by a small amount. Notice that a bit stream that contains many large frames in close succession may cause the buffer to underflow. If this bit stream is transmitted in a CBR channel, the picture data may not be received in time to be displayed. This example demonstrates the need for rate control. Rate control is primarily achieved by varying the quantization parameter used in coding the DCT coefficients.

The MPEG syntax requires the buffer size to be specified in the sequence header; thus, it is specified once at the beginning of the bit stream, and it cannot be changed for the remainder of the sequence. MPEG also requires a vbv_delay parameter to be specified in each picture header; vbv_delay indicates the length of time the picture start code must be stored in the buffer before it is decoded.

The buffer usage of an MPEG bit stream is often represented by plotting the buffer occupancy as a function of time or frame number. Figure 9 shows the decoder-buffer occupancy plots of two video sequences coded with the same CBR MPEG coder. The dashed lines represent the upper and lower bounds on the buffer size. The corresponding peak signal-to-noise ratio (PSNR) plots are shown to the right. Note that although the same video coder was used for the two sequences, the video quality and buffer usage are quite different. It is also important to note that different encoders can code the same video sequence in different ways and still produce MPEG-compliant bit streams. This is due to the wide range of coding options that can be adjusted in the video coding algorithm.

MPEG-2 Profiles and Levels. A large number of applications are addressed by MPEG, each with a number of desired functionalities. As a result, no single application is likely to use more than a small subset of the total functionalities. MPEG, therefore, grouped together appropriate subsets of functionalities and defined a set of *profiles* and *levels.* A profile corresponds to a set of functionalities (or tools) that are useful for a particular range of applications. Specifically, a profile defines a subset of the video syntax and functionalities. Currently, the profiles include (1) Main, the baseline profile for digital television, (2) Simple, a low-cost alternative to Main that does not use B frames, (3) 4 : 2 : 2, which is useful for television production, three scalable profiles (4) SNR, (5) Spatial, (6) High, and (7) a Multiview profile, which provides for stereo video. Within a profile, a level defines the maximum range on some of the parameters, such as resolution, frame rate, bit rate, and buffer size (which is a lower bound). Currently, there are four levels (1) Low, (2) Main for conventional television resolutions, (3) High-1440, and (4) High for HDTV resolutions. A decoder is specified by the profile and level that it conforms to [e.g., Main Profile at Main Level (MP@ML)]. In general, a more complex profile/level is a superset of a less complex profile/level. Two profile/levels that are likely to be widely used are Main Profile at Main Level, which can be used to compress conventional television (e.g., NTSC or PAL), and Main Profile at High Level, which can be used to compress HDTV.

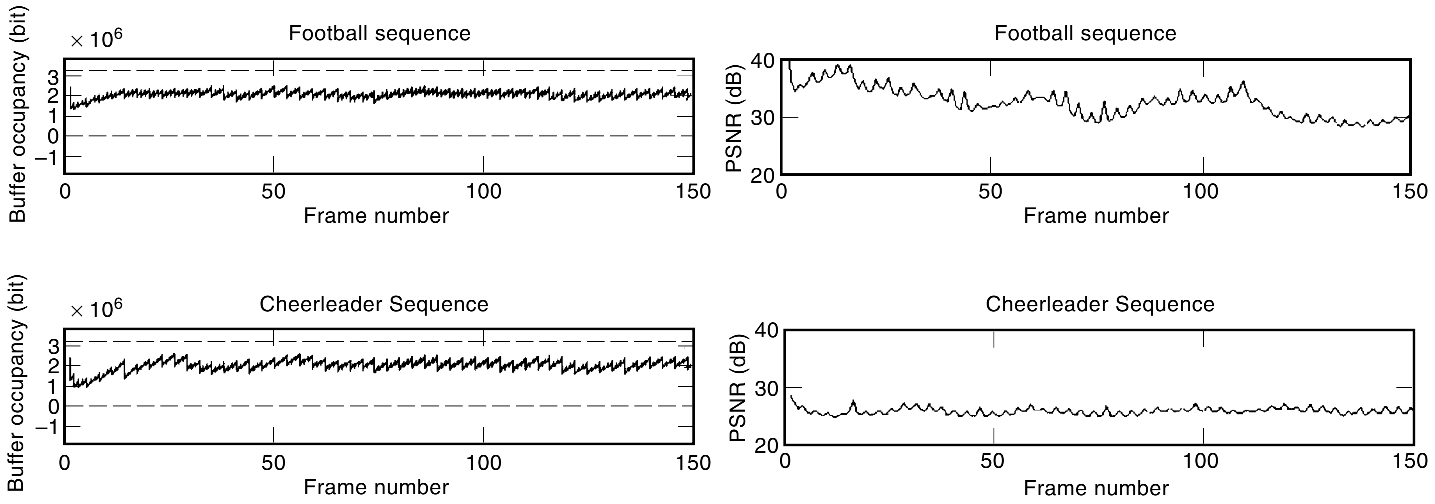

Figure 9. The buffer occupancy and PSNR as a function of frame number for two MPEG-coded sequences. The sequences are coded with a GOP pattern of *IBBPBBPBBPBBPBB*. The GOP structure is evident in the plots.

H.261 and H.263 Video Compression Standards

A number of standards were developed in the 1980s for videoconferencing, where the first to gain widespread acceptance was the ITU (CCITT) H.320 standard (10). H.320 encompassed a number of video, audio, multiplexing, and protocol standards, where the video compression standard is H.261 (11). H.261 was designed for videoconferencing over the integrated services digital network (ISDN); therefore, it is often referred to as $p \times 64$ because it is designed to operate at rates of $p \times 64$ kb/s where $p = 1, 2, \ldots, 30$. H.261 was adopted as a standard in 1990. In 1993, the H.324 standard was initiated with primary goal of videotelephony over the public switched telephone network (PSTN) (conventional analog telephone lines) (12). This corresponds to all video, audio, and control data within approximately 33.6 kb/s. The video compression portion of the standard is H.263, and its first phase was adopted in 1996 (13). An enhanced H.263, known as H.263+ because it is H.263 *plus* additional functionalities, is scheduled to be finalized in 1998 (14). A long-term version of H.263 (which may be a completely new algorithm) is scheduled for standardization in 1999. This section continues with a brief overview of the H.261, H.263, and H.263+ video compression standards.

H.261. In order to facilitate interoperability between the 525 line, 60 fields/s, and the 625 line, 50 field/s television standards, a new video format was adopted. The common intermediate format (CIF) is progressive (noninterlaced) 352×288 pixel, 30 frames/s. This format has half the number of active lines of the 625/50 television signal and the frame rate of the 525 television system, thereby simplifying communication between people using the two television systems. To facilitate low bit rate coding an additional video format QCIF was specified which has one-quarter the resolution of CIF (half the number of samples horizontally and vertically).

H.261 is a MC-DCT based algorithm, similar to MPEG, but developed a number of years before; therefore, it was the precursor to MPEG. The goal was to create a video compression standard for real-time two-way communication. Therefore, a short delay was a critical feature, and a maximum allowable delay of 150 ms was specified. H.261 uses I and P frames (no B frames) and employs 16×16-pixel ME/MC-P and 8×8-pixel Block DCT. The motion estimation is computed to full-pel (integer) accuracy and the search range is ±15 pixels. H.261 uses a RGB to YCbCr color space conversion followed by filtering and down-sampling the chroma components by 2×2, so each macroblock consists of four 8×8 luminance blocks and two 8×8 chrominance blocks. This is equivalent to MPEG's 4 : 2 : 0 format.

The compressed bit stream is a hierarchical data structure that consists of a picture layer, which is divided into several group of blocks (GOB) layers, where each GOB consists of 33 macroblocks, each composed of six 8×8 blocks of pixels.

Each macroblock can be coded in a variety of modes including intra, inter without MC (equivalent to a zero-valued MV), inter with MC. There is also the option to apply a 3×3 lowpass filter within the feedback loop to smooth the 8×8 blocks in the previous reconstructed frame. Note that a loop filter is not used in MPEG-1/2 or H.263 because they use half-pixel MC-P, and the spatial interpolation that is performed has a similar effect as the loop filter.

The DCT coefficients are quantized, zigzag scanned, and run-length coded, and each run-length of consecutive zero-valued coefficients followed by a nonzero coefficient (run, value) is Huffman coded. The quantization can be altered for each macroblock, and also all zero blocks in a inter-coded macroblock are identified and efficiently communicated to the decoder.

H.263. The H.263 video compression standard was designed with the primary goal of communication over conventional telephone lines. Transmitting video, speech, and control data over a 33.6 kb/s modem means that typically there is only about 20 kb/s to 24 kb/s available for the video.

The H.263 coder is a MC-DCT coder similar in structure to H.261, and it was designed to facilitate interoperability between H.261 and H.263 coders. A number of enhancements over H.261 were designed to (1) reduce the overhead information required, (2) improve the error resilience, (3) provide enhancements to some of the baseline coding techniques (including half-pixel MC-P), and (4) include four advanced coding options. The advanced coding options are negotiated in that the encoder and decoder communicate to determine which options can be used before compression begins. The four advanced coding options are briefly discussed.

The *unrestricted motion vector mode* allows motion vectors to point outside the actual picture area (unlike in MPEG and H.261 where the vectors are constrained to point inside), thereby providing improved prediction in cases where there is movement around the boundary. This is of special concern for small picture sizes where any inefficiencies at the boundaries can have drastic effects on the total performance.

The *advanced prediction mode* enables (1) the use of four motion vectors for the four 8 × 8 pixel blocks in a 16 × 16 macroblock instead of a single motion vector for the macroblock, (2) overlapped block motion compensation (OBMC) where the prediction of each pixel is formed by a linear combination of three predictions given by the current motion vector and two neighboring motion vectors, and (3) use of unrestricted motion vector mode. These techniques provide improved prediction and the OBMC also leads to a subjectively more appealing (smoother) video.

The *syntax-based arithmetic coding mode* enables the use of arithmetic coding instead of Huffman coding, providing a slight reduction in bit rate for the same image quality.

The *PB-frame mode* exploits some of the advantages of *B* frames as used in MPEG. Specifically, a *PB* frame consists of two frames coded as one unit, where one frame is a *P* frame and another is a *B* frame, which is predicted from the currently coded *P* frame and the last previously coded *P* frame. In general, H.263 *B* frames do not perform as well as MPEG *B* frames because, unlike MPEG, bidirectional motion vectors are not explicitly transmitted, and only a portion of each macroblock is bidirectionally predicted. However, for relatively simple video as often occurs for videophones, PB-frames perform well by increasing the frame rate while requiring only a small increase in the bit rate.

When comparing H.263 using all the coding options with H.261, H.263 typically achieves approximately a 3 dB improvement at the same bit rate, or 50% reduction in bit rate for the same signal-to-noise ratio (SNR) (quality).

H.263+. H.263+ is an extension of H.263, which includes several new features that provide improved compression performance, support for packet networks and error-prone environments, and support for a wider range of video formats.

The compression improvements include prediction of low-frequency horizontal and vertical DCT coefficients among neighboring blocks, new zigzag-type scanning patterns, new variable length code table for intra-blocks, deblocking filter mode to reduce the blocking artifacts, improved *PB* frame mode where a complete motion vector can be transmitted for *B* blocks, increased motion vector range, and a number of improvements in the quantization. A particular novel improvement is the Reference Picture Resampling mode where the reference picture used for prediction can be resized, translated, or warped before being used as the prediction for the current frame. This mode enables efficient coding of some global motions such as translation, more efficient switching between different spatial resolutions, and in general more flexibility for performing compression.

Support for packet networks and error-prone environments is provided by three types of bit stream scalability (temporal, spatial, and SNR), additional tools for partitioning a compressed bit stream into packets and later reassembling the bit stream, the ability to define independent subpictures for coding in order to limit potential error propagation, and the ability to communicate to the encoder which frames were received and decoded accurately by the decoder and therefore may be used as an accurate reference for subsequent prediction.

EMERGING VIDEO COMPRESSION STANDARDS

The MPEG committee is currently in the process of developing a new audio visual standard, commonly referred to as MPEG-4. While the primary goals of MPEG-1 and -2 were high-quality compression and communication of (natural) video and audio, the goal of MPEG-4 is a single framework for unifying how content is created, distributed, accessed, and digested within the separate but blending fields of digital television, interactive graphics applications and the World Wide Web (15).

MPEG-4 attempts to represent video as a collection of (hopefully) meaningful arbitrarily shaped visual objects, as opposed to a stream of pixels. This representation enables processing and interaction with the video based on its content, that is with each of the individual objects. MPEG-4 is developing features for supporting and integrating both natural and synthetic (computer-generated) material, providing flexibility in compositing natural and synthetic video and audio objects to form a scene as well as facilitating interaction with the content of a scene (e.g., the individual audio/visual objects) and enabling the reliable delivery of this content over heterogenious networks and error-prone environments.

Although the conventional video standards represent video using motion and pixel values, MPEG-4 represents the scene as being composed of a number of (potentially) arbitrarily shaped objects, each of which is represented using motion, texture (pixel values), shape, and compositional information. Most of the conventional video compression tools such as block-based ME/MC-P, block DCT, quantization, rate control, and Huffman coding are still important ingredients of MPEG-4. There is also considerable research toward developing novel tools for facilitating the desired functionalities (16).

The first version of MPEG-4 is scheduled to be finalized in early 1999, and a second phase incorporating additional tools is currently in progress. Also in collaboration with MPEG-4 is a "long-term" version of H.263, sometimes referred to as H.263L, which is scheduled for standardization in 1999. The final algorithm for H.263L may be completely different from the current H.263 algorithm.

MPEG is also in the process of developing another standard, referred to as MPEG-7, whose goal is to enable fast and efficient searching for multimedia content in much the same way that the conventional internet search engines enable fast searches for textual information (17–19). Specifically, the

"Multimedia Content Description Interface" standard will define a set of descriptors that can be used to describe various types of multimedia information, such as still images, video, graphics, speech, audio, and information about the creation and composition of these elements. Even though previous MPEG standards defined new compression standards, MPEG-7's goal is to represent the information describing the content, and not the content itself. Associating content description data with multimedia will facilitate fast, efficient searches, as well as indexing and general multimedia management. MPEG-7 is scheduled to be finalized in 2001.

SUMMARY AND ADDITIONAL TOPICS

This article examined the current and emerging video compression standards. These standards specify the syntax and the decoding process of the compressed bit stream. Even though these aspects of a video compression system are specified, many areas remain open enabling considerable flexibility in the system design. A few of these areas are discussed below.

These video compression standards do not specify the encoding process or the pre- and postprocessing that may be applied to the video. Thus, considerable freedom is left to the system designer. Manufacturers may choose to make any number of tradeoffs as long as the encoders produce standard-compliant bit streams and the decoders can decode standard-compliant bit streams. As a result, advancements made in video encoding algorithms (e.g., improved bit allocation strategies) may be used as long as the resulting bit stream is standard-compliant. This freedom leads to a competitive marketplace for manufacturers where there can be a wide range of price/performance video compression products.

Another important area is the development of efficient video encoder and decoder implementations for a variety of platforms and applications. For example, software-only video decoding, and in a few cases encoding, is now possible on some conventional personal computers and workstations thanks to the incorporation of multimedia operations into the general-purpose processor architectures (2). A prime example of this is the single-instruction multiple-data (SIMD) type operations for simultaneously processing a number of 8 or 16 bit data elements using longer (32 or 64 bit) processing units (e.g., the MMX instruction set for Intel's ×86 processors). In addition, considerable work remains for developing efficient low-power implementations for portable devices.

Efficient and reliable transmission and storage of compressed video is another important area. This requires detailed knowledge of the channel characteristics of the particular communication system. For example, packet networks such as the internet exhibit packet loss, whereas wireless video applications exhibit both isolated and bursty data loss. Both the H.263+ and MPEG-4 standardization efforts are examining these issues. In addition, it may be useful for the compressed bit streams to be stored such that the video can be easily browsed. This leads to the areas of scalable video compression and content-based retrieval.

Finally, as video technologies progress to compressed video environments, it may be necessary to develop efficient methods for performing conventional video operations on compressed video bit streams. That is, it may be desirable to perform some processing (such as reverse play or splicing), where both the input and output are compressed bit streams. Many of these video-processing operations are considered simple when applied to uncompressed video; however, they are much more complicated when applied to compressed video. One method for performing these tasks is to decompress the bit stream, process the reconstructed video frames, and recompress the result. This has two disadvantages. First, it is wasteful in terms of computational complexity and memory requirements. Second, generation losses may occur when recompressing the decoded video frames. Developing efficient algorithms for processing compressed video is therefore another important area of research (20,21).

BIBLIOGRAPHY

1. A. Netravali and B. Haskell, *Digital Pictures, Representation, and Compression.* New York: Plenum, 1988.
2. V. Bhaskaran and K. Konstantinides, *Image and Video Compression Standards: Algorithms and Architectures,* Boston: Kluwer, 1997.
3. N. S. Jayant, J. Johnston, and B. Safranek, Signal compression based on models of human perception, *Proc. IEEE,* **81**: 1385–1422, 1993.
4. J. S. Lim, *Two-Dimensional Signal and Image Processing,* Englewood Cliffs, NJ: Prentice-Hall, 1990.
5. J. G. Apostolopoulos and J. S. Lim, in M. Sezan and R. Lagendijk (eds.), *Video Compression for Digital Advanced Television Systems, Motion Analysis and Image Sequence Processing,* Boston: Kluwer, 1993, Chap. 15.
6. ISO/IEC 11172, Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbits/s, International Organization for Standardization, 1993.
7. D. J. LeGall, MPEG: A video compression standard for multimedia applications, *Commun. ACM,* **34**: 47–58, 1991.
8. J. L. Mitchell et al., *MPEG Video Compression Standard,* New York: Chapman & Hall, 1997.
9. ISO/IEC 13818, Generic coding of moving pictures and associated audio information, International Organziation for Standardization, 1996.
10. ITU-T Recommendation H.320, Narrow-band visual telephone systems and terminal equipment, International Telecommunication Union, March 1993.
11. ITU-T Recommendation H.261, Video codec for audiovisual services at p×64 kbits/s, International Telecommunication Union, March 1993.
12. ITU-T Recommendation H.324, Terminal for low bitrate multimedia communication, International Telecommunication Union, March 1996.
13. ITU-T Recommendation H.263, Video coding for low bit rate communication, International Telecommunication Union, March 1996.
14. ITU-T Draft Recommendation H.263 Version 2, H.263+, video coding for low bit rate communication, International Telecommunications Union, September 26, 1997.
15. MPEG committee, Overview of the MPEG-4 version 1 standard, Doc N2196, March 1998. Available at http://drogo.cselt.stet.it/mpeg/public/w2196.htm
16. Special Issue on MPEG-4, *IEEE Trans. Circuits Syst. Video Technol.,* **7** (1): 1997.
17. MPEG Requirements group, MPEG-7: Context and objectives (version 7—San Jose), Doc N2207, March 1998. Available at http://drogo.cselt.stet.it/mpeg/public/w2207.htm

18. MPEG Requirements group, MPEG-7: Requirements document v.5, Doc N2208, March 1998. Available at http://drogo.cselt.stet.it/mpeg/public/w2208.htm

19. MPEG Requirements group, MPEG-7: Applications document v.5, Doc N2209, March 1998. Available at http://drogo.cselt.stet.it/mpeg/public/w2209.htm

20. S. F. Chang and D. G. Messerschmitt, Manipulation and compositing of MC-DCT compressed video, *IEEE J. Selected Areas Commun.,* **13**: 1–11, 1995.

21. S. J. Wee and B. Vasudev, Splicing MPEG video streams in the compressed domain, *Proc. IEEE Int. Conf. Multimedia Signal Processing,* Princeton, NJ, 1997, pp. 225–230.

Reading List

Many of the video compression standards discussed in this article are continuously evolving, and the best place to find up-to-date information is at the web sites of the respective standardization committees. The official MPEG web site is http://drogo.cselt.stet.it/mpeg and the official ITU site is http://www.itu.ch. A very useful nonofficial MPEG web site is http://www.mpeg.org. These sites also contain a very large number of useful links to other information sites. Information about the Adanced Television Systems Committee (ATSC) digital television standard that has been adopted in the United States can be found at http://atsc.org.

John G. Apostolopoulos
Susie J. Wee
Hewlett-Packard Laboratories

VIDEO, DIGITAL. See Digital television.

VIDEO GAMES. See Computer games.

VIDEO, INTERACTIVE. See Interactive video.

VIDEO, MULTIMEDIA. See Multimedia video.