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

IMAGE AND VIDEO CODING

Images and video have become ubiquitous in our everyday life. Examples are consumer products from traditional television, still and video cameras, to the newer *digital* still and video cameras, modern printers and fax machines, videoconferencing systems, Kodak Photo-CD, direct broadcast satellites (*DBS*), digital video discs (*DVD*), and high-definition TV (*HDTV*). Furthermore, the Internet has made more and more images and video available. Images and video are also crucial in many applications in medicine (*CT*, *MRI*, *PET*, etc.), education (distance learning), defense (video transmission from unmanned vehicles), and space science (transmission of images and video of Mars from artificial satellites to earth). In most cases, the representation of images and video is digital.

The main things we want to do with images and video are to transmit; to store and retrieve; to manipulate (editing, authoring, etc.); and to visualize. In dealing with images and video, in most cases the amount of data (in bits) is huge. Thus, *compression* is essential to make implementation possible. For example, straight digitization of HDTV-format signals yields more than 1 Gbit/s. It is impossible to transmit this through a standard TV broadcast channel of 6 MHz bandwidth. Fortunately, compression techniques (*MPEG-2*) reduce the bit rate to 20 Mbit/s. Then, by using appropriate channel coding and modulation methods, the signal can be transmitted through a 6 MHz channel. Similarly, movies on CD, DBS, DVD, and so on are simply not possible without image/video compression.

We have come a long way from the early days (1960s) of basic research in image and video coding to the standardization activities of the last decade (and beyond). The image/video compression standards have had tremendous impact on the development of consumer products. Without the standards, the basic research results (such as transform coding of images) would still be lying dormant in the archives of scholarly journals. In this article, we review the most important basic concepts and techniques of image and video coding, describe the major standards, and discuss some of the challenging research issues.

We mention in passing that image/video coding in the broad sense includes: source coding (compression), channel coding (error detection and correction), encryption, and watermarking. This article covers only compression.

Compression Framework For Still Images

Compression of video and images is to remove redundancy from their representation. Although there are a number of ways to do this, the inspiration comes, not surprisingly, from the seminal work of Claude Shannon (1), the father of information theory, who quantified the concept of compressing "optimally." Compression can be done losslessly, meaning that there is absolutely no loss of the source information. This is critical when any loss of information is catastrophic, for example, in data files. However, there is a price paid for perfect representation. Understandably, one cannot typically achieve very high compression performance (typical numbers are 2:1 or 3:1 at best). When dealing with images and visual information, there are two dominant issues. First, the sheer volume of the source dictates the need for higher compression ratios (we could not have digital television with such meager compression ratios), and secondly, perfect replication of the source is often overkill—visual information is inherently very loss tolerant.

This leads to the concept of "lossy" compression. One can quantify the amount of loss based on objective measures like mean squared error (*MSE*) (or more perceptually meaningful measures, which have not however, proved to be very analytically tractable to date). The tradeoff between the number of bits used to represent an image (rate) versus the amount of degradation which the compression entails (distortion) has been formalized by Shannon under Rate-Distortion (*R-D*) theory. Although the theoretical bounds for specific statistical classes of sources have been provided under this theory, there are two problems: (1) actual images are very complex to model statistically, and simple models do not work well; and (2) practical compression algorithms have severe operational constraints such as low complexity, low delay, and real-time issues that need to be addressed.

This underlines the need for efficient compression frameworks that offer a good compromise between performance and complexity. The frameworks that have been most successful and found their way into commercial standards for image compression are based on what is termed *transform coding.* Standard transform-based image coding algorithms consist of three stages: a linear—typically unitary—decorrelating *transform,* followed by a *quantization* stage, and finally *entropy coding* of the quantized data. A good decorrelating transform removes *linear* dependencies from the data, thus producing a set of coefficients such that, when scalar quantized, the entropy of the resulting symbol stream is reduced substantially more than if the same quantization were applied directly on the raw image data.

Initially, transform coding became popular mainly because of the introduction of the block discrete cosine transform (*DCT*) which is the workhouse of commercial image compression standards (see next section). There are three reasons for the success of the block DCT: (1) it does a reasonably good job of compacting most of the image energy into a small fraction of the DCT coefficients; (2) because of the use of a block structure, it facilitates spatially adaptive compression, needed when dealing with the highly varying spatial statistics of typical images; (3) it has a fast implementation algorithm, based on a simple extension of the popular fast Fourier transform (*FFT*), which revolutionized digital signal processing.

Quantization. When the DCT is computed for a block of pixels, the value of its transform coefficients must be conveyed to the decoder with sufficient precision. This precision is controlled by a process called *quantization.* Basically, quantizing a DCT coefficient involves dividing it by a nonzero positive number called the *quantization step size* and rounding the quotient to the nearest integer. The bigger the quantization step size, the lower the precision of the quantized DCT coefficient, and the fewer the number of bits required to transmit it to the decoder. Usually, higher frequency coefficients are quantized with less precision, taking advantage of the fact that human eyes are less sensitive to high-frequency components in the image.

Entropy Coding. After quantization, the final step in the encoding process is to remove redundancy in the quantized symbol stream representing the quantized DCT coefficients. Because of the high-energy compaction property of the DCT, a large fraction of the transform data is quantized to zero. It is certainly unwise to assign as many bits to the zero value as it is to a nonzero value, for the same reasons that in a Morse code, it is unwise to assign the same pattern to an "e" as it is to an "x." Thus, the symbols generated are highly unbalanced in their probabilistic distributions. Some of them appear very frequently, and some others occur only rarely. It is possible to spend fewer bits (on average) to represent all possible outcomes of a certain symbol by assigning shorter binary strings to highly probable outcomes and longer strings to less probable ones. The final stage, called *entropy coding,* exploits this idea. The fundamental amount of information present was dubbed "entropy" by Shannon (1). There are many practical entropy-coding schemes (including the famous Morse code with which we are all familiar). The two most popular schemes in image coding are *Huffman coding* and *arithmetic coding.* These schemes work by different mechanisms but share the same basic idea of coding a symbol with a number of bits inversely related to the probability that it occurs. Therefore a very challenging problem for entropy coding is to generate an accurate profile for symbol probabilities. This is called the *modeling problem.* Modeling is done in two different ways: nonadaptive, in which the profile is determined empirically, and remains unchanged throughout coding, and adaptive, in which the profile is refined dynamically in the coding process by using previously coded data. The adaptive way usually produces a more accurate coding model and consequently better compression, but it require extra computational time to generate the model.

Jpeg–Still-Image Compression Standard

JPEG (pronounced "jay-peg") stands for Joint Photographic Experts Group. JPEG is a very successful still-image compression standard which is also known as ISO 10918 standard. JPEG is a compression standard for digital images having single (e.g., gray-scale images) or multiple components (e.g., color images) with each component's sample precision of 8 or 12 bits for lossy compression mode and 2 bits to 16 bits for lossless mode. JPEG is "color-blind" in the sense that the choice of coordinate system for color representation does not affect the compression process. Although there are many different features provided by the standard, here we concentrate only on the most important. Additional information can be found in the excellent reference (2) which also contains the Draft International Standard ISO DIS 10918-1.

JPEG standard specifies several encoding and decoding processes which can be combined in three main functional groups:

The "Baseline System" This group must be supported by hardware and software JPEG coders, and it represents transform-based lossy compression in *sequential mode*. In this mode, data is decoded in "one pass," as shown in Fig. 1 (compare to the progressive mode in the next group). "Baseline" JPEG has been used almost exclusively up to 1996 in many commercial applications.

The "Extended System" This group provides better compression performance at the price of somewhat increased complexity by using improved entropy-coding techniques, for example, arithmetic coding and extended Huffman coding. It also allows several types of progressive modes of operation. In contrast with the sequential mode, the progressive mode allows a coarse representation of an encoded image which is later refined as additional bits are decoded. For example, if the compressed image is transmitted over a slow link, this feature allows "previewing" the content of the data at a lower quality or resolution during transmission. This progressive feature is illustrated for *DCT-based progression* (original size but reduced quality for a lower resolution image), and for *hierarchical progression* (original quality but reduced size for a lower resolution image) in Fig. 1.

Lossless Coding This group provides lossless data compression, and it does not use transform and quantization. No information is lost during compression. The Lossless JPEG mode is based on forming predictions for a sample value from previously encoded samples and encoding only the difference between the predicted and actual values.

The first two groups represent the lossy class of compression processes, and they include transform, quantization, and entropy coding. In the following, we describe the main features of these three operations for the sequential mode. The sequential lossy encoding process is illustrated in Fig. 2.

DCT Transform. In lossy mode, JPEG decomposes each image component into 8×8 blocks for transform, quantization, and entropy coding. Blocks are processed in a raster scan order.

Basically, the DCT is a method of decomposing a block of data into a weighted sum of certain fixed patterns. Each such pattern is called a basis, and each basis is associated with coefficient representing the contribution of that pattern in the block to be coded. There are altogether 64 basis functions to completely represent any block without error, and therefore the total number of values to be coded after the transform remains unchanged. In the inverse DCT transform, each basis block is multiplied by its coefficient and the resulting 64 8×8 amplitude arrays are summed, each pixel separately, to reconstruct the original block.

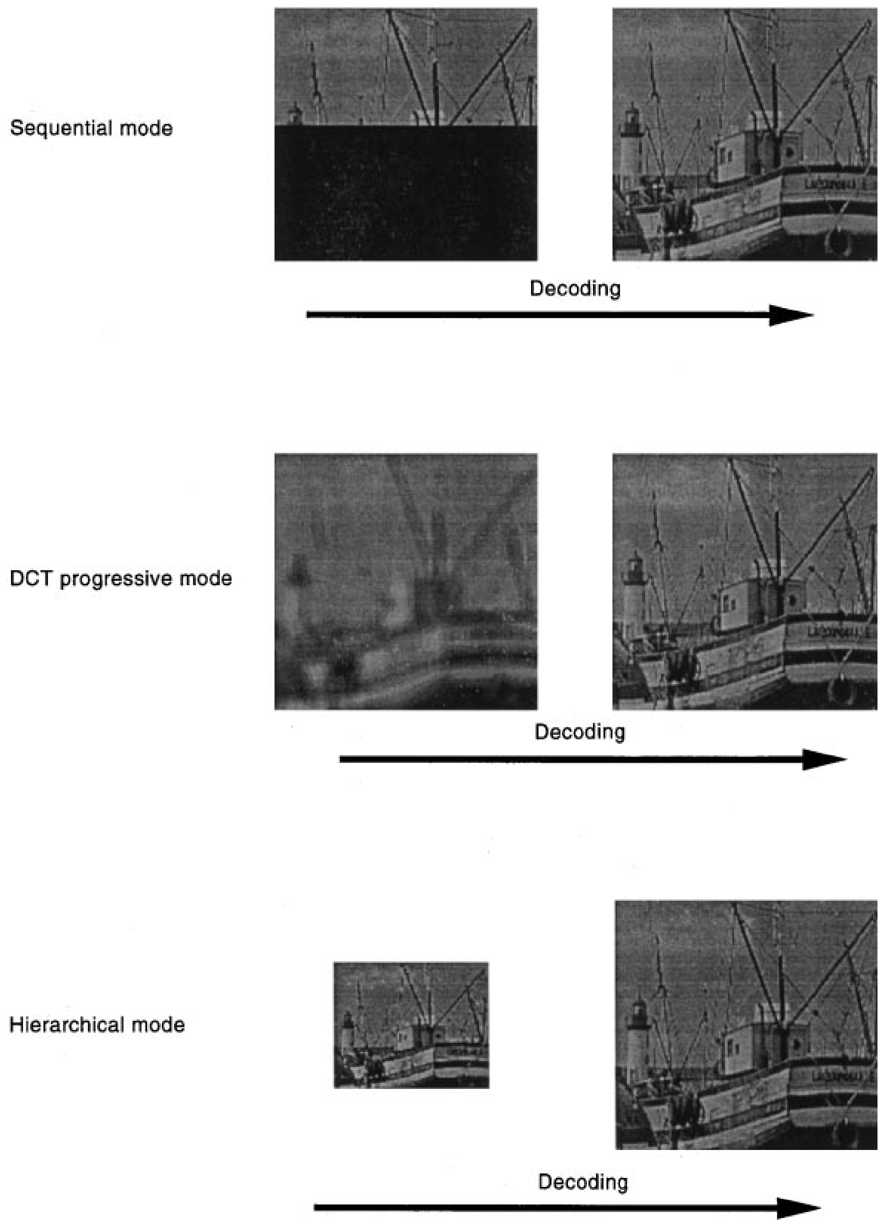

Fig. 1. Different modes of JPEG operation.

The DCT basis blocks are characterized by different spatial frequencies, a measure of how fast the pixel values change in that block. At the lowest frequency, the basis block takes a constant value. In other words, there is no variation inside it. As the frequency increases, the pixel values inside that basis block vary faster and faster. A common characteristic of all natural images and video is that low-frequency components are dominant. Therefore, after the DCT, usually only a few low-frequency coefficients are needed to approximate

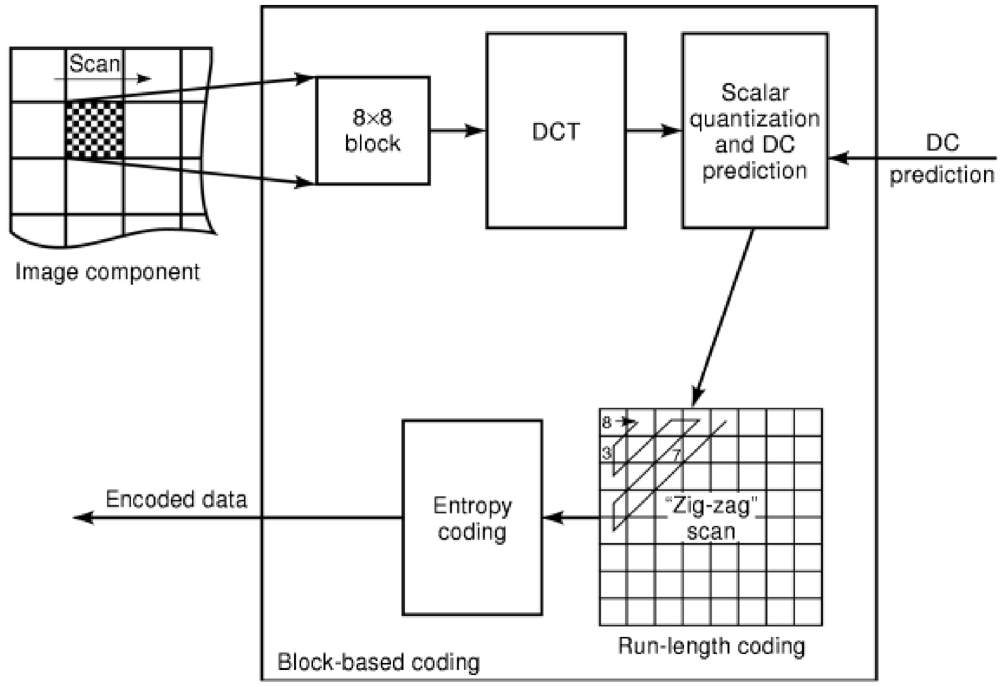

Fig. 2. Lossy sequential JPEG compression.

the original image block well, a phenomenon called *energy compaction.* The energy-compaction property makes coding the DCT coefficients much easier than directly coding the original data.

DCT is a linear orthogonal transform which can be represented as multiplication of 8 × 8 image blocks by 8 × 8 DCT basis blocks. Formally, DCT operation is described by the following equations:

$$S(v,u) = \frac{C(v)}{2}\frac{C(u)}{2}\sum_{y=0}^{7}\sum_{x=0}^{7} s(y,x)\cos\left[\frac{(2x+1)u\pi}{16}\right] \cos[(2y+1)v\pi/16]$$

where $c(x) = 1/\sqrt{(2)}$ for $x = 0$ and $C(x) = 1$ for $x > 0$. $s(y, x)$ and $S(v, u)$ are 2-D sample values and DCT coefficients, respectively.

The inverse transform is

$$s(y,x) = \sum_{v=0}^{7}\frac{C(v)}{2}\sum_{u=0}^{7}\frac{C(u)}{2} S(v,u)\cos\left[\frac{(2x+1)u\pi}{16}\right] \cos\left[\frac{(2y+1)v\pi}{16}\right]$$

Quantization. The DCT coefficients of each 8 × 8 block are quantized using a uniform scalar quantizer. Quantization step size is defined for each frequency coefficient using an 8 × 8 quantization matrix. Typically, a single quantization table is used for all components. However, up to four different tables may be used if needed.

The values of the quantization tables are encoded in the header of the compressed file. Quantization is a lossy step, that is, the information cannot be recovered perfectly at the decoder. However, the quantization which allows achieving a high compression rate at the price of some quality degradation.

Entropy Coding. The first quantized frequency coefficient, called a *DC* coefficient, represents the average sample value in a block and is predicted from the previously encoded block to save bits. Only the difference from the previous DC coefficient is encoded, which typically is much smaller than the absolute value of the coefficient. The remaining 63 frequency coefficients (called *AC* coefficients) are encoded using only the data of the current block. Because normally only a few quantized AC coefficients are not zero, their positions and values are efficiently encoded using run-length codes followed by Huffman or arithmetic entropy coders. AC coefficients are processed in a *zigzag* manner (see Fig. 2) which approximately orders coefficients from the lowest to the highest frequency. Run-length code represents the sequence of quantized coefficients as (*run, value*) pairs, where "run" represents the number of zero-valued AC coefficients between the current nonzero coefficient and the previous nonzero coefficient, and "value" is the (nonzero) value of the current coefficient. A special end-of-block (*EOB*) signals the end of nonzero coefficients in the current block. For the example in Fig. 2 with three nonzero AC coefficients, the possible sequence after the run-length encoder is (0,5)(0,3)(4,7)(EOB). The sequence of "runs" and "values" is compressed by using Huffman or arithmetic codes. The Parameters of the entropy coder are stored in the header of the compressed image.

Decoding. The decoder performs inverse operations for entropy coding and the DCT.

JPEG: Merits and Demerits. JPEG remains a popular image compression standard because of its competitive compression efficiency, low complexity and low-storage requirements, existence of efficient implementations, and the presence of several progression modes in it. However, because of its block-based coding process, JPEG-coded images suffer from the so-called "blocky artifacts," especially at high compression rates where the boundaries of the blocks become visible in the decoded image. This problem is resolved to some extent by postprocessing the decoded image. As technology improves, storage and complexity constraints for image coders become less strict and a new generation of image coders is preferred because of its improved compression efficiency. Some examples of such coders are briefly described in the following sections.

Beyond JPEG: Wavelets

Since their introduction as a tool for signal representation, wavelets have become increasingly popular within the image-coding community because of the potential gains they offer for constructing efficient image-coding algorithms (3). Those potential gains result from the fact that wavelets provide a good tradeoff between resolution in the space and frequency domains, a feature which results in mapping typical space-domain image phenomena into *structured* sets of coefficients in the wavelet domain. However, to make effective use of any structure for improving coding performance, an algorithm requires a statistical characterization of the joint distribution of wavelet coefficients, capable of taking that structure into account. A hot research topic as of the late 1990s consists of identifying properties of the wavelet coefficients of images leading to good image-coding performance.

Although wavelets share many properties with the DCT (e.g., decorrelation), and the wavelet coefficients may be quantized and entropy coded similarly to the way DCT coefficients are used, taking this same approach ignores a fundamental wavelet property, the joint time-frequency localization.

Consider the following simple image model. For coding application, typical images can be described as a set of smooth surfaces delimited by edge discontinuities (see Fig. 3). Now consider the effect of representing an edge in the wavelet domain, as illustrated by Fig. 4. The abstract property of joint time-frequency localization has a very practical consequence: the formation of energy clusters in image subbands at spatial locations associated with edges in the original image.

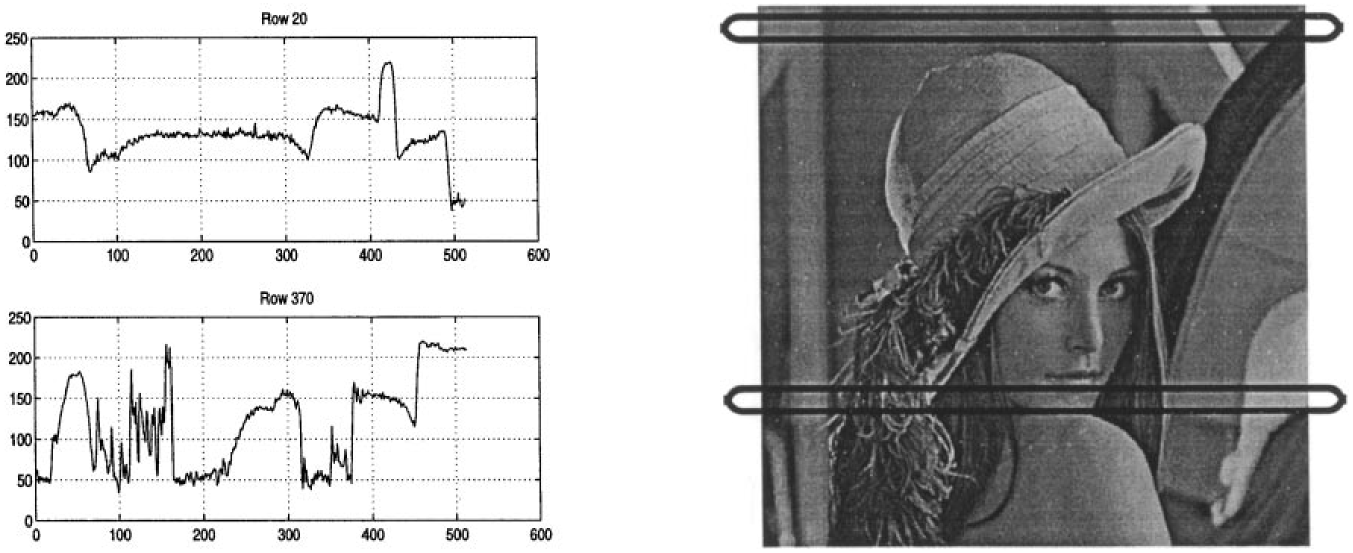

Fig. 3. Typical images can be described as a set of smooth surfaces delimited by edge discontinuities. Row 20 (up) is a mostly smooth region, whereas row 370 (down) contains some textures.

This clustering effect is attributed mostly to two basic facts: (1) signal energy due to smooth regions is compacted mostly into a few low-frequency coefficients, thus resulting in negligible contributions to coefficients in the higher frequency bands; and (2) because of the small compact support of the wavelets used, edges can contribute energy only to a small number of coefficients in a neighborhood of their space-domain location. This effect is illustrated in Fig. 5.

Now, in a data compression application, "useful" properties are those that can be described by some statistical model of the source being coded, so that a practical algorithm can exploit them to achieve better coding performance. Most state-of-the-art wavelet-based image coders are based on exploiting in an improved statistical description of image subbands, which explicitly takes into account the space-frequency localization properties of the wavelet transform.

Beyond JPEG: Fractal Coding

Fractal image coding is an image-compression method based mainly on the work of Barnsley in the 1980s (4) and that of Jacquin in the late 1980s and early 1990s (5). All image-coding methods use some form of redundancy present in images for data compression. The type of redundancy that fractal coding methods use is that of similarity across scale. This is based on the assumption that important image features, including straight edges and corners, do not change significantly under change of scale (see Fig. 6). For example, a small part of a straight edge in an image may look similar to a larger part of the same edge in that image.

Encoding. Just like any image-coding system, a fractal coding system has an encoder and a decoder. In the encoder, an image is first partitioned into small nonoverlapping blocks, called *range blocks.* For each range block, the encoder searches in the image for larger blocks that are similar to the range block and among all of them picks the one that is most similar to the range block, as shown in Fig. 7. For each range block, this selected larger block is called the *domain block.* By most similar, we mean that after applying some simple transformations on the domain block, the mean square error (*MSE*) between the range block and the

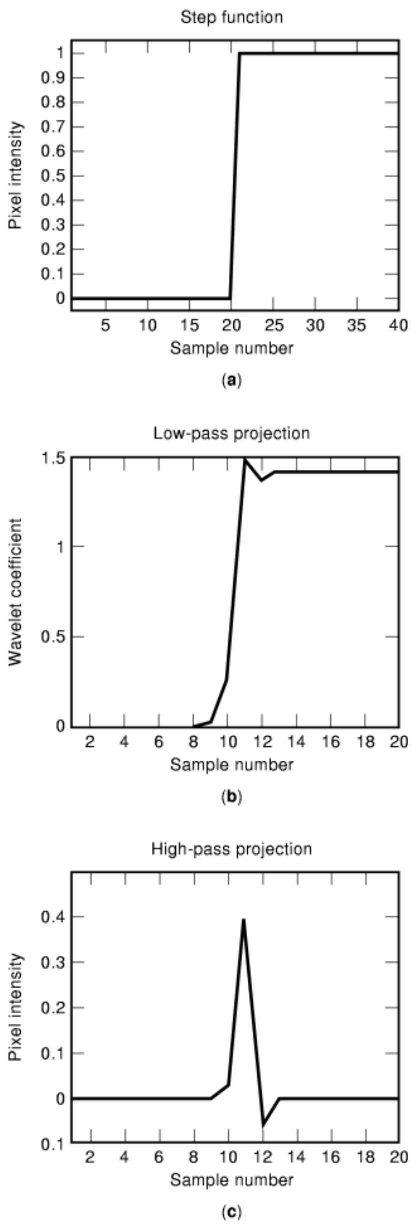

Fig. 4. To illustrate how the wavelet decompositoin preserves the locality of data characteristics in the spatial domain: (a) a step; (b) its projection on the low-pass space; (c) its projection on the high-pass space. In the high-pass projection, all of the signal energy is concentrated in a neighborhood of the spatial location of the step.

transformed domain block becomes the smallest. In addition to shrinking, which needs to be applied to a domain block to make it the same size as the range block, the transformations typically used are adding a constant value to all the pixel values in the domain block to adjust its brightness; multiplying a constant value to all of the pixel values in the domain block to adjust its contrast; rotating the domain block (typically multiples of 90°), and reflecting against the horizontal, vertical, or diagonal axis.

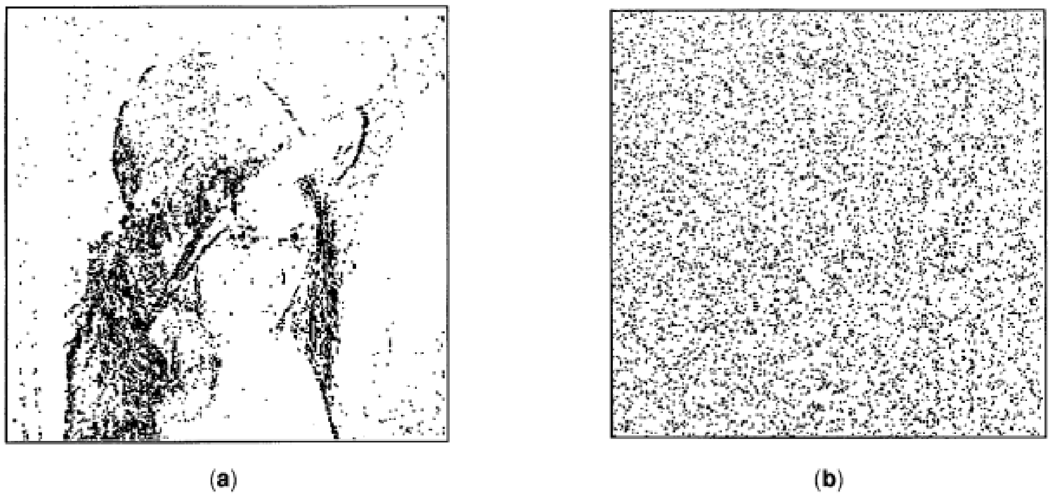

Fig. 5. To illustrate the clustered nature of wavelet data: (a) coefficients in subband LH_0 of Lena whose magnitude is above a fixed threshold (i.e., those which contain most of the signal energy necessary to reduce distortion); (b) an equal number of points as in (a), but drawn from a uniform density over the same area. This figure gives motivation for the type of techniques used later in the coder design that would be completely useless if typical realizations of wavelet data were like (b) instead of (a).

So, for each range block, the encoder needs to look at all potential domain blocks in the image and also must find the best set of transformations that may be applied on the potential domain blocks to make them similar to the range block. In practice, some range blocks may be very flat and a single DC value may be sufficient to describe them well. In such cases, the encoder may choose not to conduct the search and encode the block by simply saving its DC value. On the other hand, no good matches may be found for some blocks in the image. In such cases, the encoder may choose to break the range block into smaller range subblocks and conduct the search for the subblocks instead.

After finding the best matching domain block and its associated transformations for each range block, the encoder saves the address of the domain block and the parameters of these transformations (e.g., how much the brightness was changed, if and how much the domain block was rotated). It should be emphasized that the encoder does not save either the values of the pixels in the range blocks or the values of the pixels in the domain block. The addresses and the quantized transformation parameters for all the range blocks in the image are entropy-coded to generate the final binary code for the image. We call this binary code the *fractal code* of the image. This code represents how each part of the image is approximated by transforming other larger parts of the same image.

Now, if we go to the location of its domain block for every range block in the image, extract the block from the image, apply the corresponding transformations on it, and substitute the range block with this transformed block, we obtain a new image, which is an approximation of the original image. If we denote the original image by $\mathbf{x}_o$, and denote this operation, that is applied on $\mathbf{x}_o$, by $\mathbf{T}$, then

$$\mathbf{T}(\mathbf{x}_o) \approx \mathbf{x}_o$$

that is applying $\mathbf{T}$ on $\mathbf{x}_o$ *almost* keeps it unchanged. Note that the transformation $\mathbf{T}$ is fully represented by the fractal code.

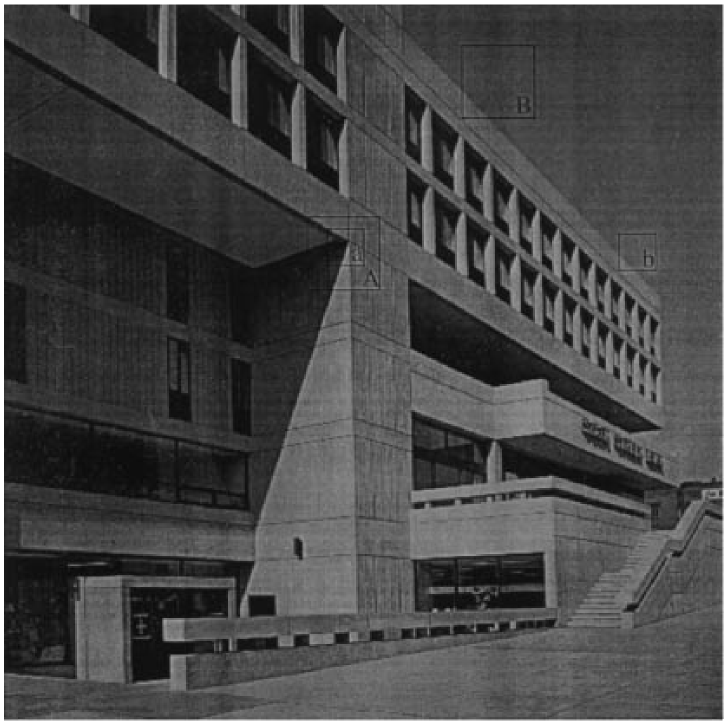

Fig. 6. Fractal coding uses the similarity of different-sized image blocks for coding. It approximates each image block with another larger image block in the same image. For example, blocks "A" and "B" in this image are very similar to blocks "a" and "b," accordingly, and can be used to approximate them.

If there is an image $\mathbf{x}_f$ such that $\mathbf{T}(x_f)$ is exactly the same as $\mathbf{x}_f$ for this transformation $\mathbf{T}$, that is,

$$\mathbf{T}(\mathbf{x}_f) = \mathbf{x}_f$$

then $\mathbf{x}_f$ is called the *fixed point* of $\mathbf{T}$.

Decoding. So far, we know how the encoder takes an image and makes a fractal code, or equivalently, a transformation $\mathbf{T}$, for it. Now the question is how this information can be used by the decoder to reconstruct the original image or an approximation of it.

According to a theorem called the *Contraction Mapping Theorem* (4), if $\mathbf{T}$ is a *contractive* transformation, then applying $\mathbf{T}$ on any arbitrary image repeatedly transforms that image to the fixed point of $\mathbf{T}$. More precisely, for any image $\mathbf{y}$,

$$\lim_{n\to\infty} \mathbf{T}^n(\mathbf{y}) = \mathbf{x}_f$$

So, the decoder can find $\mathbf{x}_f$ by simply taking any arbitrary image and applying the transformation $\mathbf{T}$, which is represented by the fractal code, on the image many times. Of course, $\mathbf{x}_f$ is usually not exactly the same as $\mathbf{x}_o$, but according to another theorem called the *Collage Theorem* (4), the closer $\mathbf{T}(\mathbf{x}_o)$ is to $\mathbf{x}_o$ the closer $\mathbf{x}_f$ will be to $\mathbf{x}_o$. In other words, if the encoder can find very good matches for range blocks, the decoder can duplicate the original image very closely. However, because the decoded image is not exactly the same as the original image, fractal coding falls in the class of lossy compression methods.

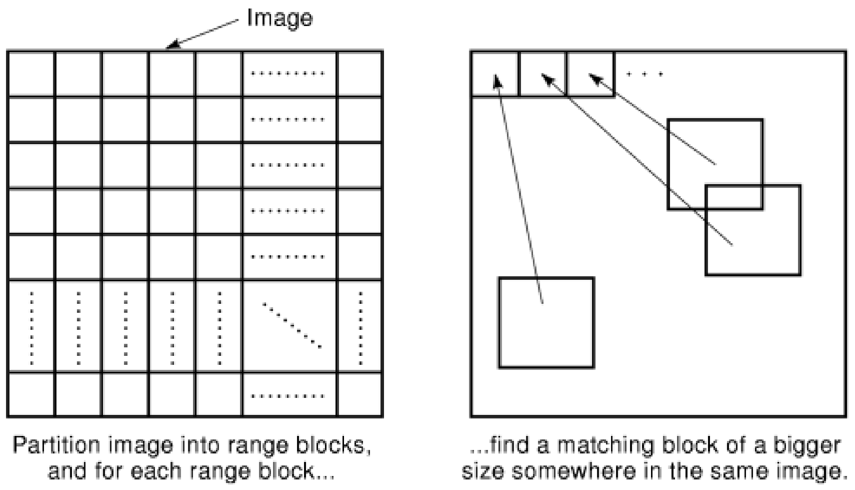
Image
Partition image into range blocks, and for each range block...
...find a matching block of a bigger size somewhere in the same image.

Fig. 7. A basic fractal image coding algorithm.

Computational Complexity. One major difficulty with fractal image coding is that the encoding process needs a lot of computation and requires a large amount of computer time. The main computations are trying every potential domain block, finding the best set of transformations for it, and computing how well the result matches the range block. However, there are techniques that alleviate this computational burden to some extent (6).

The description given previously provides the basic mechanism of fractal coding. Many different variations of this algorithm have been proposed by different researchers. Interested readers may look at (6) for more details.

Compression Framework For Video

Now we move from images to video signals. A video signal can be thought of as a sequence of images, or more correctly image *frames.* These frames are highly correlated in time, hardly surprising considering that typical video sequences come from camera shots of scenes. Thus, in addition to exploiting the spatial redundancy in individual video frames (as in still images), it is important for a good video-coding algorithm to exploit the much more significant temporal redundancy across frames. In state-of-the-art video coders, this is done by what is termed *motion-compensated prediction*. The basic idea is simple. Because little changes between successive frames of typical video scenes (e.g., when these represent 1/30 intervals), with high likelihood one can predict the next frame based on the previous or the future frame. Aiding in this prediction, it is necessary to model the motion of areas of the picture. State-of-the-art commercial video-coding algorithms send the motion of image blocks from frame to frame.

Some parts of the pictures—those in which no significant changes occur—are simply copied from adjacent pictures. Other parts may be best predicted by parts of the neighboring frames corresponding to the same objects that are displaced by motion, and this requires the use of motion compensation. All of the standards covered in the next section (MPEG-1, MPEG-2, H.261, and H.263) use a block-based algorithm for motion compensation, in which each video frame is partitioned into small rectangular blocks, and for each block, a match into a certain neighborhood in the adjacent (previous or next) frames is conducted, according to a certain

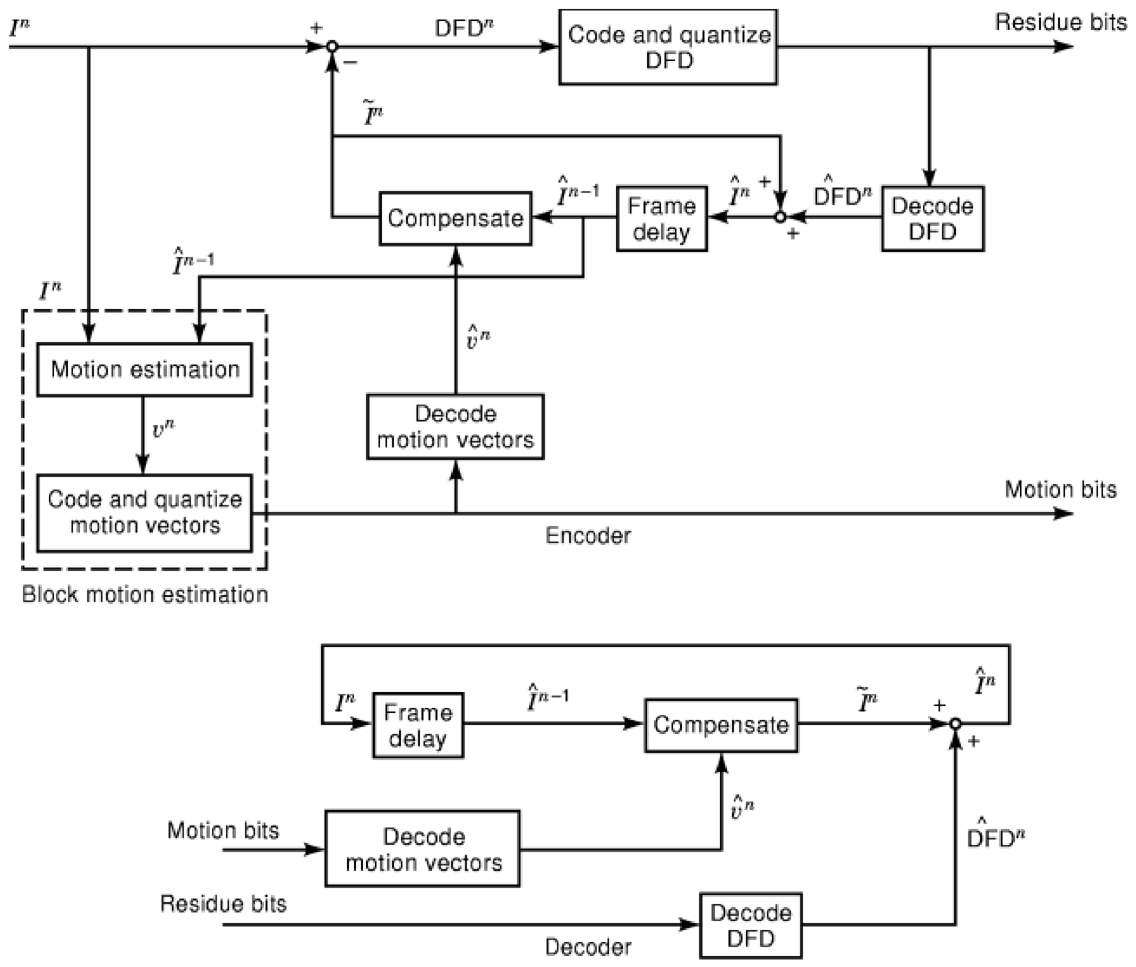

Fig. 8. Typical motion-compensated video encoder and decoder. Here I^n, $\tilde{I}^n$ and $\hat{I}^n$ represent the original, motion-compensated, and final reconstructed frame n, respectively.

error criterion (e.g., minimum mean absolute error), and the displacement to the best matching location is coded as the *motion vector.*

Clearly, after motion compensation, the compensated difference is much smaller than the original frame difference, but because one cannot always expect to predict perfectly (if one could, there would be no information to send!), one incurs prediction error. Then this error is sent. The better the prediction, the lower this error, and the fewer the number of bits needed to send it.

In current commercial video coders, the error signal (termed the *displaced frame difference* or *DFD*) is coded so that is almost identical to the way still images are coded using the JPEG standard, that is using a DCT-based framework. At the decoder, the current frame is reconstructed based on: (1) the past frame; (2) the motion information that was sent relating the previous frame to the current frame; and (3) the prediction error (which is sent in the DCT domain). This very simple framework works very well in practice, and forms the core of commercial standards, like MPEG and H.263 (see next section). Figure 8 captures the essence of this video-coding framework, which is dubbed hybrid motion-compensated video coding.

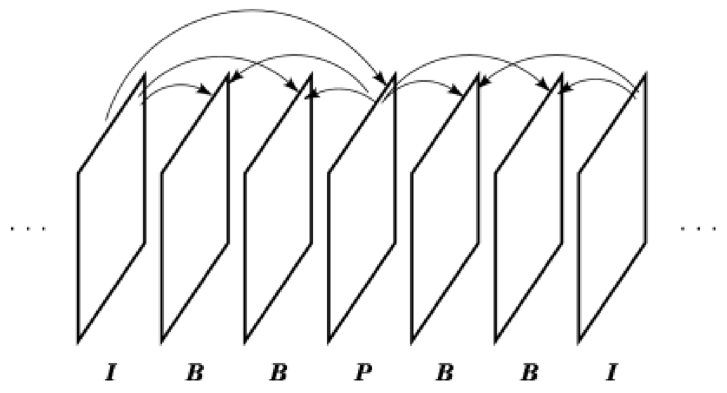

Fig. 9. The I-frames, P-frames, and B-frames in a GOP and their predictive relationships.

Video Standards

In this section, we briefly introduce the popular video-coding standards developed by MPEG and ITU-T. *MPEG* is the acronym for Moving Picture Expert Group, which was formed under the auspices of the International Organization for Standardization (*ISO*) and the International Electrotechnical Commission (*IEC*) and is devoted specifically to standardizing video-compression algorithms for the industry (7). Since it was established in 1988, it has experienced two phases, MPEG-1 and MPEG-2, and is now in its third phase, MPEG-4. Data rates and applications are distinguishing factors among them. MPEG-1 is designed for intermediate data rates on the order of 1.5 Mbit/s. MPEG-2 is intended for higher data rates (10 Mbit/s or more), and MPEG-4 is intended for very low bit rates (about 64 kbit/s or less). MPEG-2 and MPEG-4 have potential uses in telecommunications, and for this reason, a third major international standards organization, the International Telecommunications Union-Telecommunications Sector (ITU-T) is participating in their development. The ITU-T has produced two low-bit-rate video-coding standards so far, namely, H.261 and H.263 (8,9).

Frame Types and GOP. A video sequence is made up of individual pictures, also called frames, that occur at fixed time increments. Although there are many facets in video coding, perhaps the most important one is how to take advantage of the similarity between successive pictures in the sequence. The approach taken by many video coders is that most of the pictures are coded in reference to other ones, and only a few of them are coded independently. These independently coded frames are given the name "intra coded frames (*I-frames*)." The I-frames are needed to start coding the whole sequence and also to facilitate video editing and random access. Typically, I-frames are inserted into the sequence at regular time intervals. The pictures between each pair of successive I-frames are called a "Group of Pictures (*GOP*)," and are treated as a single unit in coding operations.

However, coding an I-frame is expensive because it is treated as completely "fresh" information. The high compression ratio needed in many video applications is achieved by coding other frames in a GOP as differences relative to their neighboring frames. These frames are called "predicted (P)-frames", or "bidirectionally predicted (B)-frames", depending on whether only the past frame or both the past and future frames are used as the reference, respectively. Because of the dependency between I-frames, P-frames, and B-frames, the order in which they are coded has to be carefully arranged. Typically, the I-frame in each GOP is coded first, followed by the P-frames, which are predicted from the I-frame, and then finally the B-frames, as shown in Fig. 9.

P-frames and B-frames are coded in a variety of ways. Some parts of the pictures—those in which no significant changes occur—are simply copied from adjacent pictures. Other parts may be best predicted from parts of the neighboring frames corresponding to the same objects that are displaced by motion by using the motion-compensation technique.

Both intra-and interframe coding operate on a 8×8 block basis and are facilitated by the discrete cosine transform.

After quantization, the final step in the encoding process is to entropy code all the relevant information, namely, motion vectors, quantized DCT coefficients, and control information (for instance, quantization step sizes, frame types, and so on). Huffman coding and arithmetic coding are adopted in MPEG-1/2, H.261, and H.263 for entropy coding. Although adaptive entropy coding usually produces a more accurate coding model and consequently better compression compared to nonadaptive entropy coding, the extra computation time required to update the model is not desirable in most video applications.

Differences among Various Standards. Although all of these standards center around the basic framework of block-based motion compensation and DCT coding, they are differ from each other in many significant ways.

As mentioned earlier, a major difference between MPEG-1/2 and H.261/3 is their targeted bit rates and applications. The applications targeted by MPEG are (1) digital storage, such as compact disc (*CD*), and digital video disc (*DVD*); (2) digital satelite broadcasting; (3) high-definition television (*HDTV*), and those targeted by H.261 and H.263 are low bit rate applications, like videoconferencing and videotelephony. As applications vary, the types of video sequences being coded, and the requirements for coding quality and additional functionalities also differ. For example, in videoconferencing where simple "head and shoulder" sequences are typical, attention is usually focused on the face of the speaker and less on the background parts in the picture, which therefore requires that the face region is coded with higher quality.

MPEG-2 includes all features of MPEG-1 with enhancements that improve its coding efficiency and add more flexibility and functions to it. One such example is *scalability,* which allows various quality reconstructions by partially decoding the same bit stream. Another example is *error resilience,* which enables the system to operate in a more error-prone environment. H.263 can also be regarded as an enhancement and an extension of H.261. The two most notable innovations in H.263 are the overlapped-block motion-compensation (*OBMC*) mode, in which motion-compensation accuracy for each motion block is improved by using a weighted linear prediction from neighboring blocks, and the PB frame mode, in which pairs of a P-frame and a B-frame are coded jointly to save bit rates. There are many other differences, however, in nearly every aspect of the coder, which are beyond the scope of this section.

MPEG-4 Standard

Purpose. The MPEG-4 standard was developed in response to the growing need for a coding method that facilitates access to visual objects in natural and synthetic moving pictures and associated natural or synthetic sound for various applications, such as digital storage media, the Internet, and various forms of wired or wireless communication (10). The use of this standard means that motion video can be manipulated as a form of computer data and can be stored on various storage media, transmitted and received over existing and future networks, and be distributed on existing and future broadcast channels. Target applications include, but are not limited to, such areas as Internet multimedia, interactive video games, interpersonal communications (videoconferencing, videophone, etc.), interactive storage media (optical disks, etc.), multimedia mailing, networked database services (via ATM, etc.), remote emergency systems, remote video surveillance, and wireless multimedia.

Object-Based Coding Scheme. Audiovisual information consists of the coded representation of natural or synthetic objects that can be manifested audibly and/or visually. On the sending side, audiovisual information is compressed, composed, and multiplexed in one or more coded binary streams that are transmitted. On the receiver side, these streams are demultiplexed, decompressed, composed, and presented to the

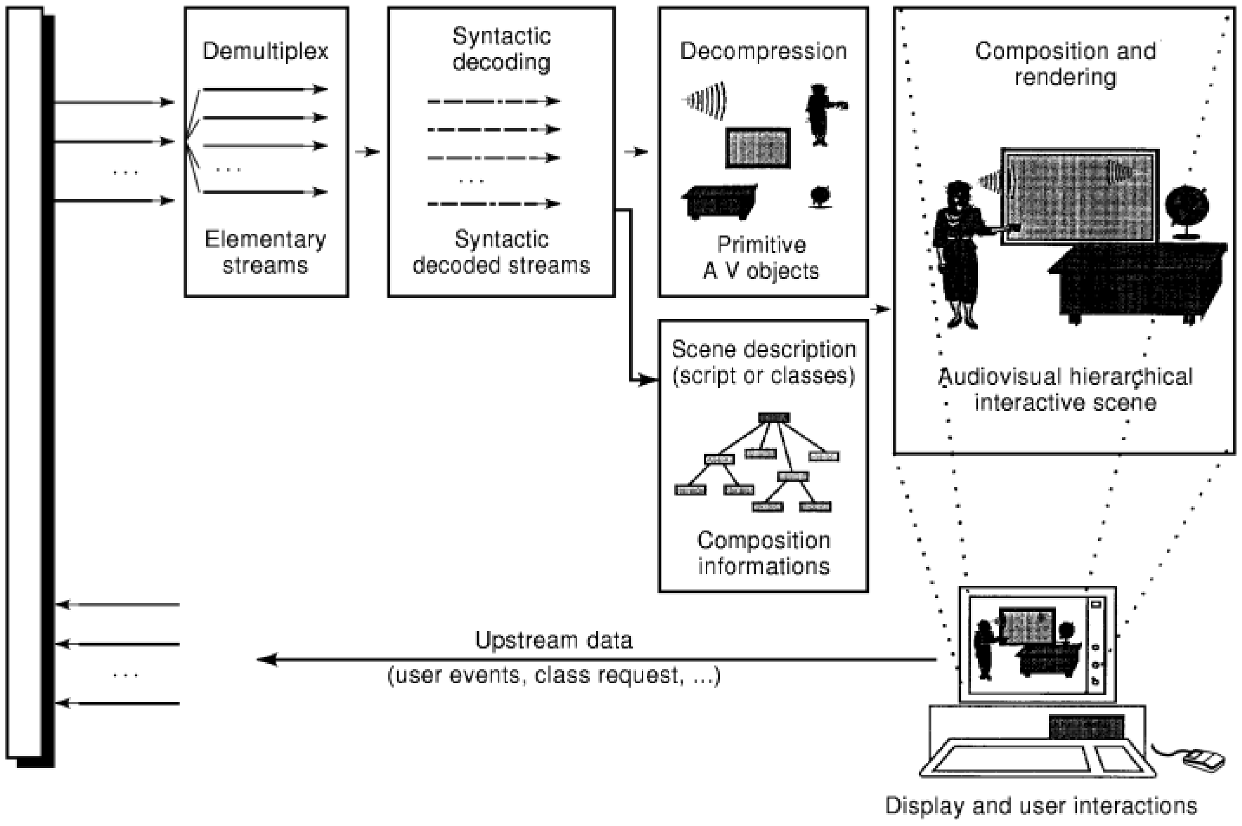
Demultiplex
Elementary streams
Syntactic decoding
Syntactic decoded streams
Decompression
Primitive A V objects
Composition and rendering
Audiovisual hierarchical interactive scene
Scene description (script or classes)
Composition informations
Upstream data
(user events, class request, ...)
Display and user interactions

Fig. 10. Processing stages in an audiovisual terminal.

end user. The end user may have the option to interact with the presentation. Interaction information can be processed locally or transmitted to the sender. The presentation can be performed by a stand-alone system or by part of a system that needs to utilize information represented in compliance with the MPEG-4 standard. In both cases, the receiver is generically called an *audiovisual terminal* or just *terminal* (Fig. 10).

Scene Description. Scene description addresses the organization of audiovisual objects in a scene in terms of both spatial and temporal positioning. This information allows composing individual audiovisual objects after they have been rendered by their respective decoders. The MPEG-4 standard, however, does not mandate particular rendering or composition algorithms or architectures. These are considered implementation-dependent. The scene description can be accomplished by using two different methodologies: parametric [Binary Format for Scenes (*BIFS*)] and programmatic [Adaptive Audiovisual Session (*AAVS*) format]. The parametric description is constructed as a coded hierarchy of nodes with attributes and other information (including event sources and targets). The scene description can evolve over time by using coded scene description updates. The programmatic description consists of downloaded Java code that operates within a well-defined execution environment in terms of available systems services for accessing decoder, composition, and rendering resources.

Interactivity. To allow active user involvement with the audiovisual information presented, the MPEG-4 standard provides support for interactive operation. Interactive features are integrated with the scene description information that defines the relationship between sources and targets of events. It does not, however, specify a particular user interface or a mechanism that maps user actions (e.g., keyboard key pressed or mouse movements) to such events.

Multiplexing/Demultiplexing. Multiplexing provides the necessary facilities for packaging individual audiovisual object data or control information related to audiovisual objects or system management and control in one or more streams. Individual components to be packaged are called *elementary streams* and are carried by the multiplexer in *logical channels.* These streams may be transmitted over a network or retrieved from (or placed on) a storage device. In addition, the multiplex layer provides system control tools, for example, to exchange data between the sender and the receiver or to manage synchronization and system buffers. The multiplex layer does not provide facilities for transport layer operation. It is assumed that these are provided by the network transport facility used by the application.

Decompression. Decompression recovers the data of an audiovisual object from its encoded format (syntax) and performs the necessary operations to reconstruct the original audiovisual object (semantics). The reconstructed audiovisual object is made available to the composition layer for potential use during scene rendering.

Profiles and Levels. Specification of profiles and levels in the MPEG-4 standard is intended to be generic in the sense that it serves a wide range of applications, bit rates, resolutions, qualities, and services. Furthermore, it allows a number of modes of coding of both natural and synthetic video to facilitate access to individual objects in images or video, called *content-based access.* Various requirements of typical applications have been considered, necessary algorithmic elements have been developed, and they have been integrated into a single syntax, which facilitates the bit-stream interchange among different applications. The MPEG-4 standard includes several complete decoding algorithms and a set of decoding tools. Moreover, the various tools can be combined to form other decoding algorithms. Considering the practicality of implementing the full syntax of this specification, however, a limited number of subsets of the syntax are also stipulated by “profile” and “level.”

An MPEG-4 “profile” is a defined subset of the entire bit-stream syntax. Within the bounds imposed by the syntax of a given profile, it is still possible to require very large variation in the performance of encoders and decoders depending on the values taken by parameters in the bit stream.

“Levels” are defined within each profile to deal with this problem. A level is a defined set of constraints imposed on parameters in the bit stream. These constraints may be simple limits on numbers. Alternatively, they may take the form of constraints on arithmetic combinations of the parameters.

MPEG-4 Objects. Three types of objects supported by the MPEG-4 standard are synthetic objects, visual objects, and audio objects. Synthetic objects, such as geometric entities, transformations, and their attributions, are structured hierarchically to support bit-stream scalability and object scalability. The MPEG-4 standard provides the approach to spatial-temporal scene composition including normative 2-D/3-D scene graph nodes and their composition supported by the Binary Interchange Format Specification (*BIFS*). Synthetic and natural object composition relies on rendering performed by the particular system to generate specific pixel-oriented views of the models.

Coded visual data can be of several different types, such as video data, still-texture data, 2-D mesh data or facial animation parameter data. Audio objects include natural audio, synthetic audio, natural speech, and text-to-speech objects. The following paragraphs briefly explain the visual data object.

Video Object. A *video object* in a scene is an entity that a user is allowed to access (seek, browse) and manipulate (cut and paste). Video objects at a given time are called *video object planes* (*vops*). The encoding process generates a coded representation of a vop and compositional information necessary for display. Further, at the decoder, a user may interact with and modify the compositional process as needed.

The framework allows coding of individual video objects in a scene and also the traditional picture-based coding, which can be thought of as a single rectangular object. Furthermore, the syntax supports both nonscalable coding and scalable coding. Thus it becomes possible to handle normal scalabilities and object-based scalabilities. The scalability syntax enables reconstructing useful video from pieces of a total bit stream. This is achieved by structuring the total bit stream in two or more layers, starting from a stand-alone base layer and adding a number of enhancement layers. The base layer can use the nonscalable syntax and can be coded using nonscalable syntax, or in the case of picture-based coding, even a different video-coding standard.

To ensure the ability to access individual objects, it is necessary to achieve a coded representation of its shape. A natural video object consists of sequence of 2-D representations (at different time intervals) called vops here. For efficient coding of vops, both temporal redundancies and spatial redundancies are exploited. Thus a coded representation of a vop includes representing its shape, its motion, and its image contents.

Still-Texture Object. Visual texture, herein called still-texture coding, is designed to maintain high visual quality in the transmission and render texture under widely varied viewing conditions typical of interaction with 2-D/3-D synthetic scenes. Still-texture coding provides a multilayer representation of luminance, color, and shape. This supports progressive transmission of the texture for image buildup as it is received by a terminal. Downloading the texture resolution hierarchy for constructing image pyramids used by 3-D graphics APIs is also supported. Quality and SNR scalability are supported by the structure of still-texture coding.

Mesh Object. A 2-D *mesh object* is a representation of a 2-D deformable geometric shape, with which synthetic video objects may be created during a composition process at the decoder by spatially piecewise warping existing video object planes or still-texture objects. Instances of mesh objects at a given time are called *mesh object planes* (*mops*). The geometry of mesh object planes is coded losslessly. Temporally and spatially predictive techniques and variable-length coding are used to compress 2-D mesh geometry. The coded representation of a 2-D mesh object includes representing its geometry and motion.

Face Object. A 3-D (or 2-D) *face object* is a representation of the human face that is structured to portray the visual manifestations of speech and facial expressions adequate to achieve visual speech intelligibility and the recognition of the mood of the speaker. A face object is animated by a stream of *face animation parameters* (*FAP*) encoded for low-bandwidth transmission in broadcast (one-to-many) or dedicated interactive (point-to-point) communications. The FAPs manipulate key feature control points in a mesh model of the face to produce animated visemes for the mouth (lips, tongue, teeth) and animation of the head and facial features, like the eyes.

A simple FAP stream can be transmitted to a decoding terminal that animates a default face model. A more complex session can initialize a custom face in a more capable terminal by downloading *face definition parameters* (*FDP*) from the encoder. Thus specific background images, facial textures, and head geometry can be portrayed.

The FAP stream for a given user can be generated at the user's terminal from video/audio, or from text-to-speech. FAPs can be encoded at bit rates up to 2 kbit/s to 3kbit/s at necessary speech rates. Optional temporal DCT coding provides further compression efficiency in exchange for delay. Using the facilities of systems, a composition of the animated face model and synchronized, coded speech audio (low-bit-rate speech coder or text-to-speech) can provide an integrated low-bandwidth audio/visual speaker for broadcast applications or interactive conversation.

MPEG-7 and Multimedia Databases

MPEG-1 and -2 deal with coding at the pixel level. MPEG-4 codes the information at the object level. MPEG-7, a work item newly started in October 1996, will code information at the content level. From MPEG-1 to MPEG-7, the abstraction level of coding increases. Because MPEG-7 is still at its early stage, the description of MPEG-7 in this section is based on the currently available MPEG documents (10). It is highly possible that the final MPEG-7 standard may differ from that described here.

MPEG-7 is formally called *Multimedia Content Description Interface*. It is motivated by the observation that current multimedia data lack a standard content-descriptive interface to facilitate retrieval and filtering. Nowadays, more and more audiovisual information is available all around the world. But before anyone can use this huge amount of information, it has to be located first. This challenge leads to the need for a mechanism that supports efficient and effective multimedia information retrieval. In addition, another scenario is associated with broadcasting. For instance, an increasing number of broadcast channels are available, and this makes it difficult to select (filter) the broadcast channel that is potentially interesting.

The first scenario is the pull (search) application and the second is the push (filtering) application. MPEG-7 aims to standardize a descriptive interface for the multimedia content so that it can easily be searched and filtered. Specifically, MPEG-7 will standardize the following:

- a set of descriptive schemes and descriptors
- a language to specify descriptive schemes, that is a Description Definition Language (*DDL*)
- a scheme for coding the description

Before we go any further into MPEG-7, it is important first to introduce some key terminologies used throughout the MPEG-7 description:

- Data Audiovisual information that will be described by using MPEG-7, regardless of the storage, coding, display, transmission, medium, or technology. This definition is intended to be sufficiently broad to encompass video, film, music, text and any other medium, for example, an MPEG-4 stream, a video tape, or a book printed on paper.
- Feature A feature is a distinctive part or characteristic of the data which stands for somebody for something in some respect or capacity (10). Some examples are the color of an image, the style of a video, the title of a movie, the author of a book, the composer of a piece of music, the pitch of an audio segment, and the actors in a movie.
- Descriptor A descriptor associates a representational value with one or more features. Note that it is possible to have multiple representations for a single feature. Examples might be a time code to represent duration or both color moments and histograms to represent color.
- Description Scheme The Description Scheme (*DS*) defines a structure and the semantics of the descriptors and/or description schemes that it contains to model and describe the content.
- Description A description is the entity that describes the data and consists of a descriptive scheme and instantiation of the corresponding descriptors.
- Coded Description Coded description is a representation of the description allowing efficient storage and transmission.
- Description Definition Language This is the language to specify Description Schemes. The DDL will allow the creation of new DSs and extension of existing DSs.

MPEG-7, like the other members of the MPEG family, is a standard representation of audiovisual information that satisfies particular requirements. The MPEG-7 standard builds on other (standard) representations, such as analog, PCM, MPEG-1, -2, and -4. One functionality of the MPEG-7 standard is to provide references to suitable portions of other standards. For example, perhaps a shape descriptor used in MPEG-4 is also useful in an MPEG-7 context, and the same may apply to motion vector fields used in MPEG-1 and -2. MPEG-7 descriptors, however, do not depend on the ways the described content is coded or stored. It is possible to attach an MPEG-7 description to an analog movie or to a picture printed on paper.

Because the descriptive features must be meaningful in the context of the application, they will differ for different user domains and different applications. This implies that the same material can be described by using different types of features tuned to the area of application. Taking the example of visual material, a lower abstraction level would be a description, for example, of shape, size, texture, color, and movement. The highest level would give semantic information: “This is a scene with a barking brown dog on the left and a blue ball that falls down on the right, with the sound of passing cars in the background.” All of these descriptions are coded efficiently for search. Intermediate levels of abstraction may also exist.

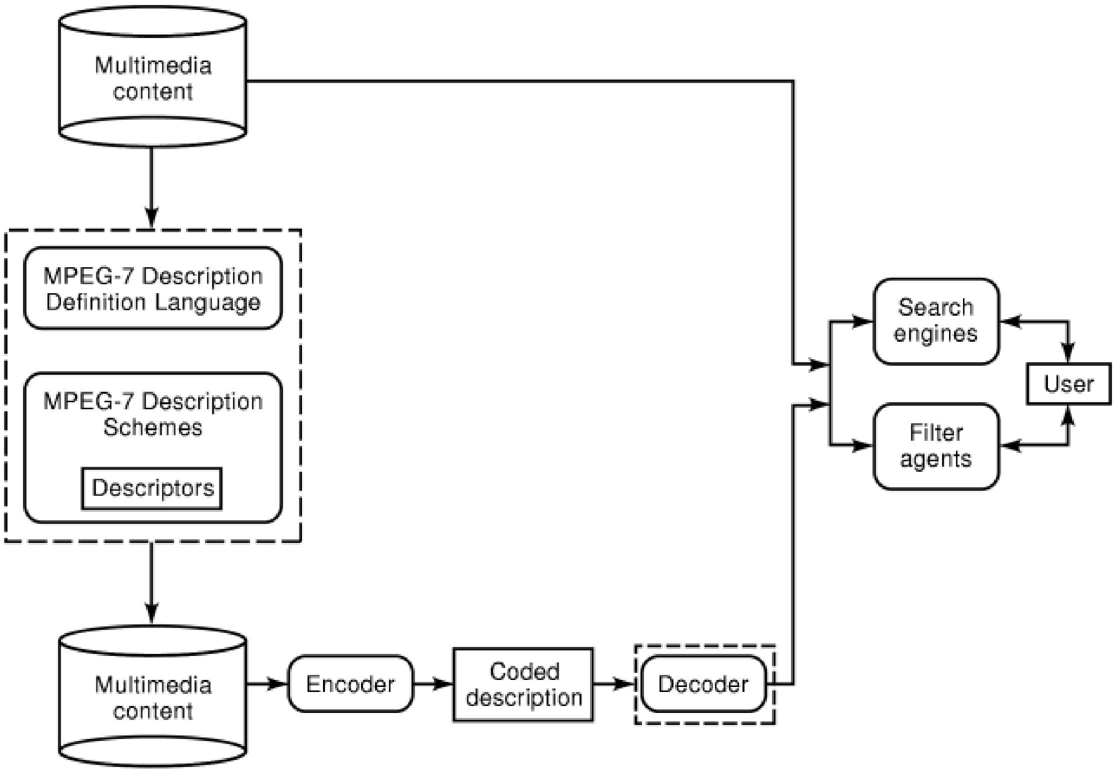

Fig. 11. A block diagram of MPEG-7 application.

MPEG-7 data may be physically located with the associated audiovisual material in the same data stream or on the same storage system, but the descriptions could also exist somewhere else on the globe. When the content and its descriptions are not co-located, mechanisms that link audiovisual material and their MPEG-7 descriptions are useful. These links should work in both directions.

Figure 11 illustrates the functionality and information flow within the MPEG-7 framework. The dotted box denotes the normative elements of the MPEG-7 standard. Note that not all the components in the figure need to be present in all MPEG-7 applications. It is also possible that there are other streams from the content to the user besides those in the figure.

Upon completing MPEG-7, many applications, both social and economic, may be greatly affected, including:

- education
- journalism (e.g., searching speeches of a certain politician using his name, his voice, or his face)
- tourist information
- entertainment (e.g., searching a game, karaoke)
- investigative services (human characteristic recognition, forensics)
- geographical information systems
- medical applications
- shopping (e.g., searching for clothes that you like)
- architecture, real estate, interior design

- social (e.g., dating services)
- film, video, and radio archives

The call for proposals of MPEG-7 will be released in October, 1998 and the proposals are due in February, 1999. The following are called for by the MPEG group:

- Description Schemes
- Descriptors
- Description Definition Language (*DDL*)
- Coding methods for compact representation of Descriptions
- Tools for creating and interpreting Description Schemes and Descriptors (not among the normative parts)
- Tools for evaluating content-based audiovisual databases (not among the normative parts)

The final International Standard is expected in September 2001.

Current and Future Research Issues

We are entering an exciting era of research in image and video coding. Many important and challenging research issues emerge from applications related to the Internet, multimedia, and virtual reality:

(1) Interplay between networking and compression: Transmission of images and video over heterogeneous computer networks (involving possibly ATM networks, wireless, etc.) introduces many research issues. These include joint source-channel coding, error-resilient coding, and quality of service.
(2) Interplay between processing and compression: For many applications, it is important to process in the compressed domain to reduce storage and processing requirements. Thus, we need algorithms for image/video analysis and understanding in the compressed domain.
(3) Interplay between multimedia database retrieval/visualization and compression: In the past, image/video compression was mainly for transmission. More and more, applications involve the storage, retrieval, and visualization of the data. Thus, the performance criteria of compression algorithms needs to be reexamined. For example, in many cases, it is important to decode at different image resolutions.

Acknowledgments

The authors would like to thank Igor Kozintsev, Yong Rui, Sergio Servetto, Hai Tao, and Xuguang Yang for their generous help in writing this article.

BIBLIOGRAPHY

1. C. E. Shannon A mathematical theory of communication, *Bell Syst. Tech. J.*, **27**: 379–423, 1948.
2. W. Pennebaker J. Mitchell *JPEG Still Image Data Compression Standard*, New York: Van Nostrand Reinhold, 1993.
3. M. Vetterli J. Kovacevic *Wavelets and Subband Coding*, Upper Saddle River, NJ: Prentice-Hall, 1995.
4. M. F. Barnsley H. Rising III *Fractals Everywhere*, 2nd ed., Boston, MA: Academic Press Professional, 1993.
5. A. E. Jacquin Image coding based on a fractal theory of iterated contractive image transformations, *IEEE Trans. Image Process.*, **1**: 18–30, 1992.
6. Y. Fisher (ed.) *Fractal Image Compression: Theory and Application*, New York: Springer-Verlag, 1995.

7. L. M. Joan *et al. MPEG Video Compression Standard*, New York: International Thomson, 1997.
8. International Telecommunication Union (ITU), *Video Codec for Audiovisual Services at p $\times$ 64 kb/s*, Geneva, Switzerland, 1993, Recommendation H.261.
9. International Telecommunication Union (ITU), *Video Coding for Low Bit Rate Communication*, Geneva, Switzerland, 1996, Recommendation H.263.
10. *MPEG Web Page* [Online], 1998. Available www: http://drogo.cselt.stet.it/mpeg/

KANNAN RAMCHANDRAN
MOHAMMAD GHARAVI-ALKHANSARI
THOMAS S. HUANG
University of Illinois at Urbana-Champaign