Image and Video Coding Standards

Rangarajan Aravind
Glenn L. Cash
Donald L. Duttweiler
Hsueh-Ming Hang
Barry G. Haskell
Atul Puri

Most image or video applications involving transmission or storage require some form of data compression to reduce the otherwise inordinate demand on bandwidth and storage. Compatibility among different applications and manufacturers is very desirable, and often essential. This paper describes several standard compression algorithms developed in recent years.

Introduction

The International Organization for Standardization (ISO) Joint Bilevel Image Group (JBIG) has perfected a progressive coding algorithm for bilevel (two-tone, black/white, or facsimile) images that transmits these images in stages of successively higher resolution. (See Panel 1 for definitions of abbreviations, acronyms, and terms.) This enables users to browse through remotely located image databases. It also allows output displays with differing resolutions to access documents that reside in the same database. New coding techniques make it possible to provide this progressive capability, while at the same time achieving significantly better compression than that attained by previous facsimile coding standards.

The ISO Joint Photographic Experts Group (JPEG) has developed an algorithm for coding single-frame color images. It is based on the discrete cosine transform (DCT), but it also has extensions for progressive coding. Starting from an original red, green, blue (RGB) picture of 24 bits per picture element (pel or pixel), the JPEG algorithms give good image quality at compression factors of 10 to 20, i.e., bit rates between 1 and 2 bits per pixel.

The International Telegraph and Telephone Consultative Committee (CCITT) Study Group 15 (SG15) and its experts group on video telephony has finalized a set of coding standards, known informally as the P×64 standard, for sending videotelephone or videoconference pictures on integrated services digital network (ISDN) facilities. The standard is applicable over a bandwidth range from 56 kilobits per second (kb/s) to 2 megabits per second (Mb/s). It relies not only on the DCT, but also on motion-compensated prediction to compress data generated by the moving imagery.

The ISO Motion Picture Experts Group (MPEG) has developed both audio and video compression algorithms that can compress entertainment or educational video for storage or transmission on various digital media, including compact disk, remote video databases, movies on demand,[1] cable television (CATV), fiber to the home, etc. Requirements are for implementation of normal play, fast forward/reverse, random access, normal reverse, and simple very-large-scale integration (VLSI). The MPEG algorithm utilizes all the P×64 methodology, as well as some new techniques, most notably conditional motion-compensated interpolation.

JBIG Progressive Bilevel Image Coding

This section presents the JBIG bilevel image coding standard and how it relates to other standards. It also describes progressive coding and compares the compression performance of various algorithms.

Standards Framework. JBIG was chartered in 1988 to establish a standard for the progressive coding of bilevel images. The "joint" in its name reflects the fact that it reports to both ISO (specifically, ISO-IEC/JTC1/SC29/WG9) and CCITT (specifically, CCITT/SGVIII/Q16). The JBIG standard[2,3] is nearly finalized.

On average, since 1988 the chair of the working group has scheduled three JBIG meetings a year, each with about 15 attendees from large, well-known companies in the fields of telecommunications, photography, and computer science.

Panel 1. Abbreviations, Acronyms, and Terms

B-frames — bidirectionally predicted, interpolative-coded frames
CATV — cable television, or community antenna television
CBP — coded block pattern
CCIR — International Radio Consultative Committee
CCITT — International Telegraph and Telephone Consultative Committee
CD-ROM — compact disk read-only memory
CIF — common intermediate format
codec — coder-decoder
CRT — cathode-ray tube
DAT — digital audio tape
DCT — discrete cosine transform
dpi — dots per inch
ECS — entropy-coded segment
EOB — end of block
EOI — end of image
FDCT — forward discrete cosine transform
FLC — fixed-length code
GBSC — group of blocks start code
GN — group number
GOB — group of blocks
GQUANT — quantizer information
HRD — hypothetical reference decoder
IDCT — inverse discrete cosine transform
IEC — International Electrotechnical Commission
I-frame — intra-coded frame
IQ — inverse quantizer
ISO — International Organization for Standardization
JBIG — Joint Bilevel Image Group
JTC1 — Joint Technical Committee 1
JPEG — Joint Photographic Experts Group
MB — macroblock
MC — motion compensation
MCU — minimum-coded unit
MPEG — Motion Picture Experts Group
MQUANT — quantizer
MVD — motion vector data
P-frames — predictive-coded frames
pixel — picture element
PSC — picture start code
PTYPE — type information
Q16 — Question 16
QCIF — quarter-CIF
RGB — red, green, blue
SC29 — Subcommittee 29
SGVIII — Study Group 8
SOF — start of frame
SOI — start of image
TR — temporal reference
VHS — Video Home System is a registered trademark of the Victor Company of Japan, Limited.
VLC — variable-length code
VLI — variable-length integer
VLSI — very large-scale integration
WG9 — Working Group 9

Relationship to Existing Standards. For bilevel image coding, the G3 and G4 algorithms[4] of CCITT Recommendations T.4[5] and T.6[6] are well established. JBIG coding, like the coding of the G3/G4 algorithms, is lossless (bit-preserving), with decoded images digitally identical to input images. Hence, image quality is not an issue using any of the available algorithms. However, compared to G3/G4 coding, JBIG coding offers better compression and, if desired, progressiveness. Numerical data relating JBIG and G3/G4 compression are discussed later in this paper. Progressive coding will be defined and discussed as well, along with identifying applications in which it is valuable.

Another standard that has applicability overlapping that of JBIG is the JPEG standard,[7] described later in this paper. Although JBIG was chartered for work on bilevel compression, the JBIG algorithm can also be used effectively for the lossless coding of grey scale images (monochrome with shades of grey) and color images. The simple expedient of letting each bit plane of such images define an independent image for bilevel coding works quite well as long as the bit planes are defined using something like a folded-binary (Gray) representation[8] of intensity. This minimizes the total number of transitions in the images of the various bit planes. When intensity resolution is highly precise and there are eight

or more bits per pixel, JBIG coding and lossless JPEG coding are about equal in compression efficiency. When the intensity resolution is coarser, JBIG coding is more efficient. Of course, if lossless coding is not required, JPEG coding in any of its normal (lossy) modes will provide the greatest compression.

The JBIG approach to lossless grey scale and color image coding offers coding unification. One underlying algorithm efficiently codes bilevel images, grey scale photographic images, color photographic images, and computer-generated images with bit-plane overlays.

Progressive Coding. Progressive codings are multiresolution encodings. An image is captured as a compression of a low-resolution rendition plus a sequence of "delta" files that each allow one doubling of resolution. When an image that has been progressively encoded is decoded, the low-resolution rendition of the original becomes available first, with subsequent doublings of resolution following as more data are decoded.

The number, D, of doublings that are to be available is a free parameter for the JBIG algorithm. When progressiveness is desired, it is typically chosen as 4, 5, or 6. It can, however, be chosen as 0, in which case progressiveness is disabled, but the JBIG compression advantage remains.

Progressive coding offers advantages for:

- Storing images in databases intended to serve displays of differing resolution capability
- Browsing through images
- Transmitting images over a packet network.

By storing progressive encodings of images, a database can efficiently serve output devices that have differing resolution capability. The database sends the coding of the low-resolution rendition and only as many delta files as needed. If a user first views an image on a comparatively low-resolution display, such as a cathode-ray tube (CRT), and later requests a hard copy on a higher-resolution display, such as a laser printer, only a few additional delta files need be sent.

In contrast, an image database storing images nonprogressively can use one of two methods to serve output terminals with different resolutions. Most simply, it can store multiple compressions at various resolutions. Alternatively, it can store only a compression at the highest resolution and require output devices to decode to this high resolution and map down to the lower resolution of the display available. The first alternative wastes storage and is inefficient when an update to higher resolution is requested. The second alternative wastes both transmission capacity and processing power. The output device must receive and decode the highest resolution rendition, even though it only can show a lower resolution rendition.

Progressive codings can be advantageous for document browsing. A low-resolution rendition can be rapidly transmitted and displayed, and then followed by as much resolution enhancement as desired. Progressive coding makes it easy for a user to recognize the image being displayed quickly and to interrupt the transmission of an unwanted image.

This advantage for progressive coding only occurs on medium-rate links, roughly those with speeds between 9.6 and 64 kb/s when bilevel images are being retrieved. Were the communication link slower, no viewer would have the patience to browse through images, no matter what the form of presentation. On high-speed links, the image is delivered so rapidly relative to human reaction times that the way it develops is immaterial.

The third application for progressive coding is in packet networks,[9] where packets can or must be classified as droppable (i.e., those that the network is free to discard during times of congestion) or nondroppable (i.e., those that the network must always deliver). The packets carrying the information for the final resolution doubling would be sent at low priority; if they had to be dropped, no image regions would be lost or destroyed. The only penalty would be an image that is slightly less sharp in some regions.

One potential disadvantage of progressive coding is its need for a frame buffer large enough to hold the image at the second-to-highest resolution. When the display is a CRT, this buffer always exists and this need is inconsequential. It is of greater concern in hard-copy devices. The JBIG algorithm has a feature called "compatible-sequential" mode, which can obviate the need for the frame buffer whenever a database is storing images progressively (to support a range of display resolutions efficiently), but can also serve hard-copy devices. For a hard-copy device, the intermediate resolution images are of no interest. In serving such a device in the compatible-sequential mode, the same information is transmitted as would be transmitted for normal progressive decoding. However, it is rearranged to eliminate the need for a full-image buffer. Reference 2 describes how this is accomplished.

Table I. Compressed File Sizes in Bytes for Various Coding Algorithms

Image	Bytes					
	Raw	G3D1	G3D2	G4	Nonprogressive JBIG	Progressive JBIG
CCITT #1	513216	37423	25967	18103	14715	16771
CCITT #2	513216	34367	19656	10803	8545	8933
CCITT #3	513216	65034	40797	28706	21988	23710
CCITT #4	513216	108075	81815	69275	54356	58656
CCITT #5	513216	68317	44157	32222	25877	28086
CCITT #6	513216	51171	28245	16651	12589	13455
CCITT #7	513216	106420	81465	69282	56253	60770
CCITT #8	513216	62806	33025	19114	14278	15227
Halftone	834048	483265	572259	591628	131479	103267

Compression Comparison. Table I shows compression performance on the eight standard CCITT test images and one additional image. The additional image is a binary image, rendering grey scale using halftoning. It is image number 20 of the so-called "JBIG testing" image set and is a picture of a Japanese woman holding flowers. The eight CCITT images are all sampled at 200 dots per inch (dpi) and contain 1728×2376 pixels. The halftone image contains 2304×2896 pixels. Compressed-file byte counts are provided for coding with one-dimensional G3 (G3D1), two-dimensional G3 (with a k factor of 4) (G3D2), G4, nonprogressive JBIG, and progressive JBIG with four delta layers.

Over the eight CCITT images, nonprogressive JBIG coding has about a 22-percent coding advantage over G4, the most efficient of the G3/G4 algorithms. The progressive JBIG algorithm provides progressivity and still shows an average 15-percent coding gain over G4.

The G3/G4 algorithms are not suitable for coding bilevel images rendering grey scale using halftoning, as is evident in the last row of Table I, where the JBIG compression advantage is about a factor of five.

Overview of JBIG Algorithm. This section describes some of the main functional blocks of an encoder. Decoders, similar to encoders, and somewhat simpler because resolution reduction is not needed, will not be described.

Conceptually, a JBIG encoder can be decomposed (see Figure 1) into a chain of D identical differential layer encoders, followed by a bottom-layer encoder. In Figure 1a, I_D denotes the image at layer D and C_D denotes its encoding. Generally, implementations will time-share one physical differential layer encoder, but for heuristic purposes, the decomposition of Figure 1a is helpful.

The heart of both the differential-layer encoder (Figure 1b) and bottom-layer encoder (Figure 1c) is an adaptive arithmetic encoder. Arithmetic coders are distinguished from other entropy coders such as Huffman coders and Ziv-Lempel coders in that, conceptually at least, they map a string of symbols to be coded into a real number on the unit interval [0.0,1.0). What is transmitted instead of the symbols is a binary representation of this number. The process to derive the representative real number is known as interval subdivision. Abramson[10] credits Elias with having conceived it soon after Shannon's seminal work on information theory was published. However, practical application of arithmetic coding had to wait almost thirty years for the discovery of ways to realize arithmetic coders with finite-precision arithmetic, as well as ways to make pipelining possible. Pipelining enables an encoder to start outputting the bits of the binary representation before it has seen the entire input stream to be coded, and for a decoder to start outputting the reconstructed symbol stream before it has seen the entire binary expansion of the representative real number. The JBIG and JPEG arithmetic coders are identical.

An algorithmic subfunction of differential-layer encoders, but not bottom-layer encoders, is resolution reduction, which is the mapping of a given resolution

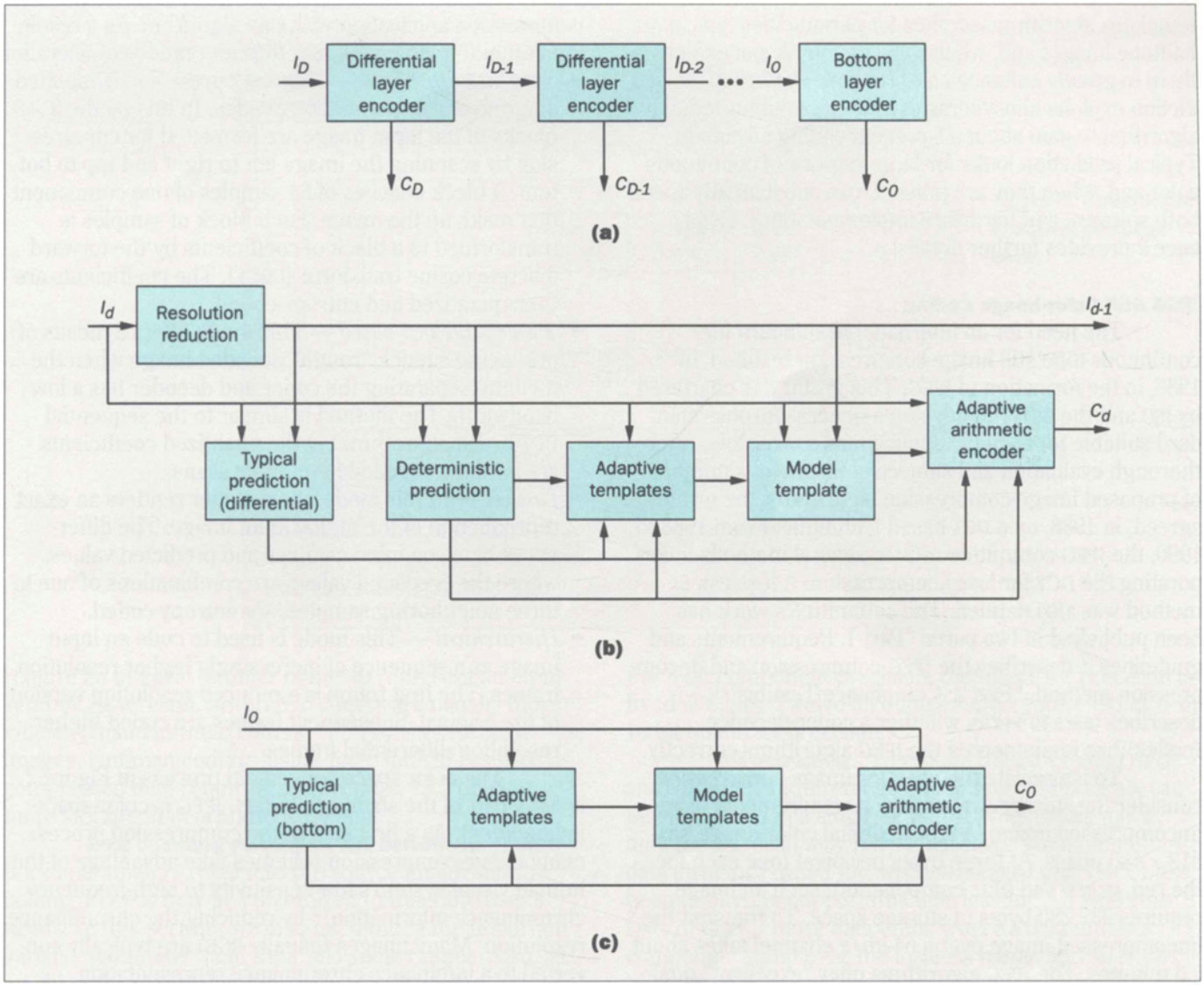


Figure 1. (a) A JBIG encoder can be decomposed into a chain of (b) *D* differential layer encoders, followed by a (c) bottom-layer encoder.

image to a half-resolution image. One way to do this would be simply to discard every other row and column, but such subsampling leads to images that are poorer in subjective quality than need be. The table-based JBIG resolution-reduction algorithm creates excellent quality, low-resolution renditions for text, line art, dithered grey scale, halftoned grey scale, and error-diffused grey scale. The low-resolution image is created pixel by pixel in the usual raster scan order, that is, from top to bottom and left to right. The color of any given low-resolution pixel is uniquely determined by the colors of nine particular high-resolution neighbors that are in fixed spatial relationship to it and three particular low-resolution neighbors that are in causal and fixed spatial relationship to it. Decoders have no counterpart to this block.

Other algorithmic subfunctions of interest are adaptive templates, deterministic prediction (differential-layer encoder only), and typical prediction. The adaptive

templates algorithm searches for periodicities typical of halftone images and, when they are found, can exploit them to greatly enhance compression. Deterministic prediction exploits idiosyncrasies of the resolution reduction algorithm to gain about a 5-percent coding advantage. Typical prediction looks for large regions of continuous color and, when they are present, can substantially speed both software and hardware implementations. Reference 2 provides further details.

JPEG Still-Color-Image Coding

The need for an international standard for continuous-tone still image compression resulted, in 1986, in the formation of JPEG. This group was chartered by ISO and the CCITT to develop a general-purpose standard suitable for as many applications as possible. After thorough evaluation and subjective testing of a number of proposed image-compression algorithms, the group agreed, in 1988, on a DCT-based technique. From 1988 to 1990, the JPEG committee refined several methods incorporating the DCT for lossy compression. A lossless method was also defined. The committee's work has been published in two parts: "Part 1: Requirements and guidelines"[7] describes the JPEG compression and decompression method. "Part 2: Compliance Testing"[11] describes tests to verify whether a coder-decoder (codec) has implemented the JPEG algorithms correctly.

To appreciate the need for image compression, consider the storage/transmission requirements of an uncompressed image. A typical digital color image has 512×480 pixels. At three bytes per pixel (one each for the red, green and blue components), such an image requires 737,280 bytes of storage space. To transmit the uncompressed image over a 64-kb/s channel takes about 1.5 minutes. The JPEG algorithms offer "excellent" quality for most images compressed to about 1.0 bit/pixel. This 24:1 compression ratio reduces the required storage of the 512×480 color image to 30,720 bytes, and its transmission time to about 3.8 seconds. Applications for image compression may be found in desktop publishing, education, real estate, and security, to name a few.

In the next section, we give an overview of the JPEG algorithms. In subsequent sections, we present some operating parameters and definitions, and describe each of the JPEG operating modes in more detail.

Overview of the JPEG Algorithms. The JPEG committee could not satisfy the requirements of every still-image compression application with one algorithm. As a result, the committee proposed four different modes of operation:

- *Sequential DCT-based* — Figure 2 presents a simplified diagram of a sequential DCT codec. In this mode, 8×8 blocks of the input image are formatted for compression by scanning the image left to right and top to bottom. A block consists of 64 samples of one component that make up the image. Each block of samples is transformed to a block of coefficients by the forward discrete cosine transform (FDCT). The coefficients are then quantized and entropy-coded.
- *Progressive DCT-based* — This mode offers a means of producing a quick "rough" decoded image when the medium separating the coder and decoder has a low bandwidth. The method is similar to the sequential DCT-based algorithm, but the quantized coefficients are partially encoded in multiple scans.
- *Lossless* — In this mode, the decoder renders an exact reproduction of the digital input image. The differences between input samples and predicted values, where the predicted values are combinations of one to three neighboring samples, are entropy-coded.
- *Hierarchical* — This mode is used to code an input image as a sequence of increasingly higher-resolution frames. The first frame is a reduced resolution version of the original. Subsequent frames are coded higher-resolution differential frames.

The color space conversion process in Figure 2 is not a part of the standard. In fact, JPEG is color-space-independent. As a first step in the compression process, many image-compression schemes take advantage of the human visual system's low sensitivity to high-frequency chrominance information[12] by reducing the chrominance resolution. Many images (usually RGB) are typically converted to a luminance-chrominance representation before this processing takes place.

Either Huffman or arithmetic techniques can be used for entropy coding in any of the JPEG modes of operation (except in the *baseline* system, where Huffman coding is mandatory). A Huffman coder compresses a series of input symbols by assigning short code words to frequently occurring symbols and long code words to improbable symbols.[13,14] The output of an arithmetic coder is a single real number. After initialization to a range of 0 to 1, the probability of each input symbol is used to restrict the range of the output number further. Unlike a Huffman coder, an arithmetic coder does not

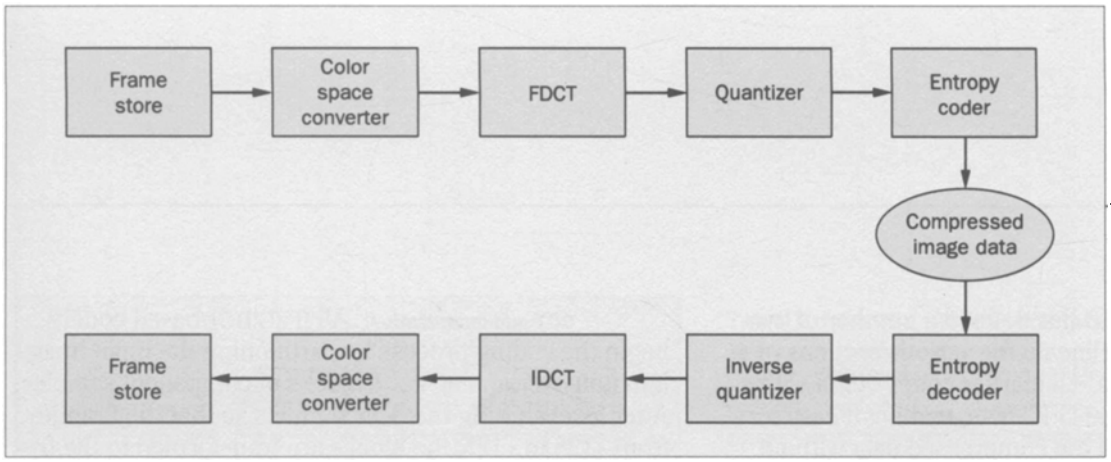


Figure 2. Sequential DCT codec.

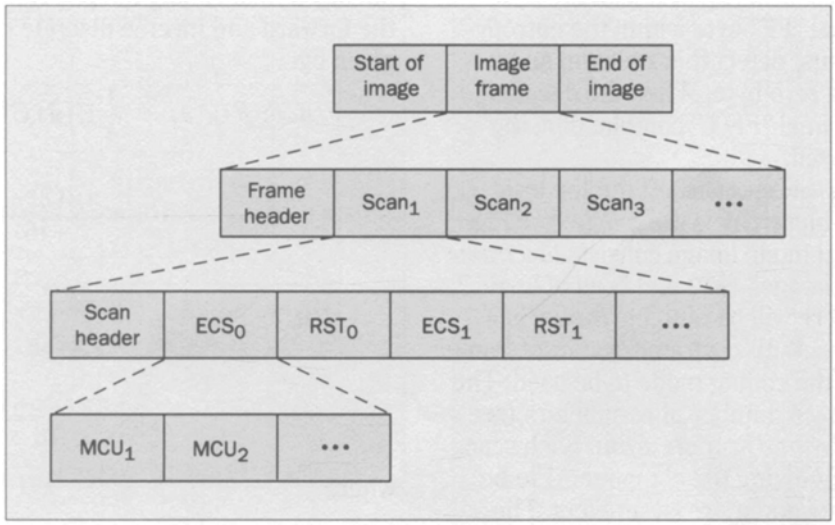


Figure 3. Structure of compressed-image data.

require an integral number of bits to represent an input symbol. As a result, arithmetic coders are usually more efficient than Huffman coders.[15,16] For the JPEG test images, Huffman coding (using fixed tables) resulted in compressed data requiring, on average, 13.2 percent more storage than arithmetic coding.

JPEG Operating Parameters and Definitions. A number of parameters related to the source image and the coding process may be customized to meet the user's needs. In this section, we discuss some of the important variable parameters and their allowable ranges. Also, as an aid to the algorithm descriptions in the following sections, we define some JPEG terms and present the hierarchical structure of the compressed data.

Parameters. An image to be coded using any JPEG mode may have from 1 to 65,535 lines and from 1 to 65,535 pixels per line. Each pixel may have from 1 to 255 components (only 1 to 4 components are allowed for progressive mode). The operating mode determines the allowable precision of the component. For the DCT modes, either 8 or 12 bits of precision are supported (only 8-bit precision is allowed for *baseline*). Lossless mode precision may range from 2 to 16 bits. If a DCT operating mode has been selected, the quantizer precision must be defined. For 8-bit component precision, the quantizer precision is fixed at 8 bits. Twelve-bit components require either 8- or 16-bit quantizer precision.

Data interleaving. To reduce the processing delay and/or buffer requirements, up to four components can be interleaved in a single scan (for progressive mode, only the DC scan may have interleaved components). A data structure called the *minimum-coded unit* (MCU) has been defined to support this interleaving. An MCU consists of one or more data units, where a data unit is a component sample for the lossless mode, and an 8×8 block of component samples for the DCT modes. If a scan contains only one component, then its MCU is equal to one data unit. For multiple component scans, the MCU for the scan consists of interleaved data units. The maximum number of data units per MCU is 10. As an interleaving example, consider an International Radio Consultative Committee (CCIR) 601 digital image in which the chrominance components are subsampled 2:1 horizontally. For a DCT coder, a CCIR-601 MCU could consist of two Y blocks, followed by a C_R block and a C_B block, where Y is the luminance of the image and C_R and C_B are proportional to the two color differences $(R-Y)$ and $(B-Y)$, respectively.

Marker codes. JPEG has defined a number of two-byte marker codes to delineate the various sections of a compressed data stream. All marker codes begin with a byte-aligned hexadecimal "FF" byte, making it easy to scan and extract parts of the compressed data without actually decoding it. Because it is possible to create a byte-aligned hexadecimal "FF" byte within the entropy-coded data, the coder must detect this situation and follow the "FF" byte with a zero byte. When the decoder encounters the hexadecimal "FF00" combination, the zero byte must be removed.

Compressed-image data structure. At the top level of the compressed data hierarchy is the *image* (see Figure 3). A nonhierarchical mode image consists of a *frame* surrounded by "start of image" (SOI) and "end of image" (EOI) marker codes. There will be multiple *frames* in a hierarchical mode image. Within a frame, a start of frame (SOF) marker identifies the coding mode to be used. The SOF marker is followed by a number of parameters (see Reference 7), and then by one or more *scans*. Each scan begins with a header identifying the components to be contained within the scan, and more parameters. The scan header is followed by an entropy-coded segment (ECS). An option exists to break the ECS into chunks of MCUs called *restart intervals* (RST_0, RST_1, etc.). The restart interval structure is useful for identifying select portions of a scan, and for recovery from limited corruption of the entropy-coded data. Quantization and entropy-coding tables may either be included with the compressed image data or communicated separately.

Sequential DCT. The sequential DCT mode offers excellent compression ratios, while maintaining image quality. A subset of the sequential DCT capabilities has been identified by JPEG for a "*baseline system.*" All DCT-based JPEG implementations are required to include baseline capability. This requirement should help to ensure interoperability between codecs from different vendors. Restrictions on the baseline system related to sample and quantizer precision were pointed out in the "Parameters" subsection. One further restriction should be noted: Although a full sequential DCT coder may employ either Huffman or arithmetic entropy coding, a baseline coder can only use Huffman coding. In addition, only two AC and two DC tables may be used per scan (up to four sets of tables are allowed for full sequential mode).

The following subsections describe the processing steps for a baseline coder. A decoder is formed by reversing the coder steps.

DCT and quantization. All JPEG DCT-based coders begin the coding process by partitioning the input image into non-overlapping 8×8 blocks of component samples. After level-shifting the 8-bit samples so that they range from –128 to +127, the blocks are transformed to the frequency domain using the FDCT.[17,18] The equations for the forward and inverse discrete cosine transforms are given by:

$$FDCT: F(u,v) = \frac{1}{4} C(u)\, C(v) \sum_{x=0}^{7} \sum_{y=0}^{7} f(x,y) \cos\frac{\pi u(2x+1)}{16} \cos\frac{\pi v(2y+1)}{16} \quad (1)$$

$$IDCT: f(x,y) = \frac{1}{4} \sum_{u=0}^{7} \sum_{v=0}^{7} C(u)\, C(v)\, F(u,v) \cos\frac{\pi u(2x+1)}{16} \cos\frac{\pi v(2y+1)}{16} \quad (2)$$

where

$$C(u), C(v) = \frac{1}{\sqrt{2}} \text{ for } u,v = 0; \; C(u)\, C(v) = 1 \text{ otherwise.}$$

The DCT concentrates most of the energy of the component samples' block into a few coefficients, usually in the top-left corner of the DCT block. The coefficient in the immediate top-left corner is called the DC coefficient because it is proportional to the average intensity of the block of spatial domain samples. The AC coefficients corresponding to increasingly higher frequencies of the sample block progress away from the DC coefficient.

The next step in the process, quantization, is the key to most of the JPEG compression. A 64-element quantization matrix, where each element corresponds to a coefficient in the DCT block, is used to reduce the amplitude of the coefficients, and to increase the number of zero-value coefficients. The quantization and dequantization is performed according to equations (3) and (4), respectively.

$$Fq(u,v) = round\left[\frac{F(u,v)}{Q(u,v)}\right] \quad (3)$$

$$R(u,v) = Fq(u,v)\, Q(u,v) \quad (4)$$

A carefully designed quantization matrix will produce

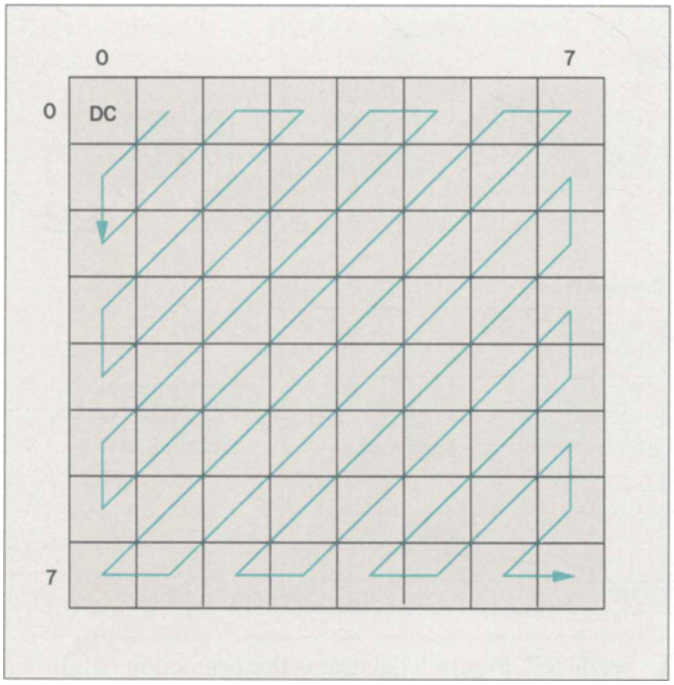


Figure 4. Zig-zag scan.

high compression ratios while introducing negligible "visible" distortion.[19] Up to four quantization matrices are allowed by JPEG. The standard does not mandate quantization matrices, but includes a set that gives good results for CCIR-601 type images. Many JPEG implementations control the compression ratio (and output image quality) by using a *q-factor*, which is usually just a scale factor applied to the quantization matrices.

DC coefficient entropy coding. Greater compression efficiency can be obtained if a simple predictive method is used to entropy-code the DC coefficient separately from the AC coefficients. Recall that the DC coefficient corresponds to the average intensity of the component block. Adjacent blocks will probably have similar average intensities. It is, therefore, advantageous to code the *differences* between the DC coefficients of adjacent blocks rather than their values. Each differential DC value is coded using a variable-length code (VLC) and a variable-length integer (VLI). The VLC corresponds to the size, in bits, of the VLI, while the VLI gives the amplitude of the differential DC value.

Zig-zag scan and AC coefficient entropy coding. After they have been quantized, the coefficient blocks usually contain many zero-value AC coefficients. If the coefficients are reordered, using the zig-zag scan illustrated in Figure 4, there will be a tendency to have long runs of zeroes. Only the nonzero AC coefficients are entropy-coded. As in the DC coefficient coding, a VLC-VLI pair results from the coding of an AC coefficient. However, the AC VLC corresponds to two pieces of information: the number of zeroes (run) since the last nonzero coefficient, and the size of the VLI following the VLC.

Table II. Lossless Mode Predictors

Selection value	Prediction
0	No prediction
1	a
2	b
3	c
4	a + b - c
5	a + ((b - c)/2)
6	b + ((a - c)/2)
7	(a + b)/2

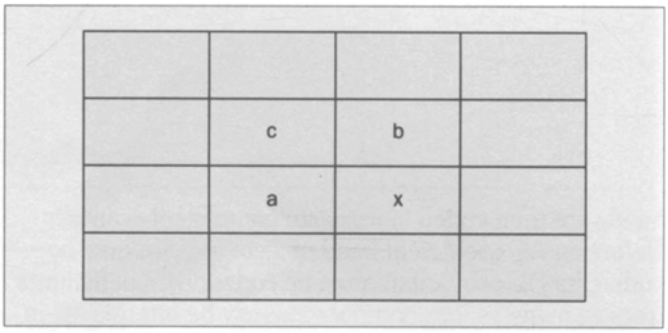


Figure 5. Prediction neighborhood.

Progressive DCT. A progressive DCT mode has been defined by JPEG to satisfy the need for a fast decoded picture when a low-bandwidth medium separates a coder and decoder. By partially encoding the quantized DCT coefficients in multiple scans, the decoded image quality builds progressively from a coarse level to the quality attainable with the quantization matrices. Either spectral selection, successive approximation, or a combination of the two is used to code the quantized coefficients.

Spectral selection. In this method, the quantized DCT coefficients of a block are first partitioned into non-overlapping bands along the zig-zag block scan. The

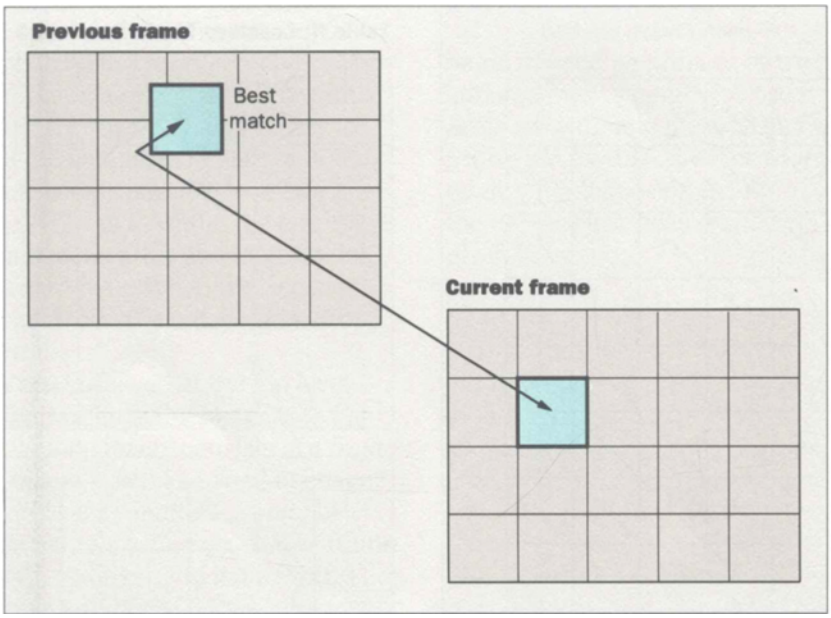


Figure 6. Block motion compensation.

bands are then coded in separate component scans. Before an AC coefficient band of a component may be coded, its DC coefficient must be coded. DC coefficients from as many as four components may be interleaved in a single scan. Interleaving is not permitted for AC bands because of the introduction of an efficient means for coding contiguous blocks of zero-valued coefficients. From 1 to 32,767 blocks can be coded with a single VLC–VLI combination called an end-of-band code.

Successive approximation. With this method, the precision of the coefficients is successively increased during multiple scans. Following a scan for a specified number of most significant bits of the quantized coefficients, subsequent scans increase the precision in increments of one bit until the least significant bits have been coded.

Lossless Mode. The lossless mode was defined for applications in which output pixels from a decoder must be identical to the input pixels to the coder. The compression ratios achievable with the lossless mode, typically around 2:1, are much smaller than those afforded by the lossy modes. This method is similar to the one used to code the DC coefficients in the DCT-based modes, but the predictor is selectable from one of seven choices, as shown in Table II. Samples a, b, and c in the table correspond to neighbors of the sample x to be predicted. Figure 5 illustrates the prediction neighborhood. Entries 1 to 3 in Table II are used for one-dimensional predictive coding, and 4 through 7 form two-dimensional predictors. Entry 0 identifies differential coding for the hierarchical mode. As in the DC coefficient entropy coding described earlier, differences between the actual and predicted values are entropy-coded.

Hierarchical Mode. In the hierarchical mode, an image is coded as a succession of increasingly higher-resolution frames. This "pyramidal" approach offers an alternative to the previously described methods for achieving progression. It also allows decoders with different resolution capabilities to use the same compressed data stream.

The first coded frame is created by reducing the resolution of the input image by a power of two in one or both dimensions, and then processing the lower resolution image using one of the lossy or lossless techniques of the other operating modes. Subsequent frames are formed by upsampling the decoded image by a factor of two in the dimension(s) having reduced resolution, subtracting the upsampled image from the input image at the same resolution, and coding the difference. "Missing" pixels in the upsampled image are filled in using linear (or bilinear) interpolation. This process continues until the decoded image has the same resolution as the

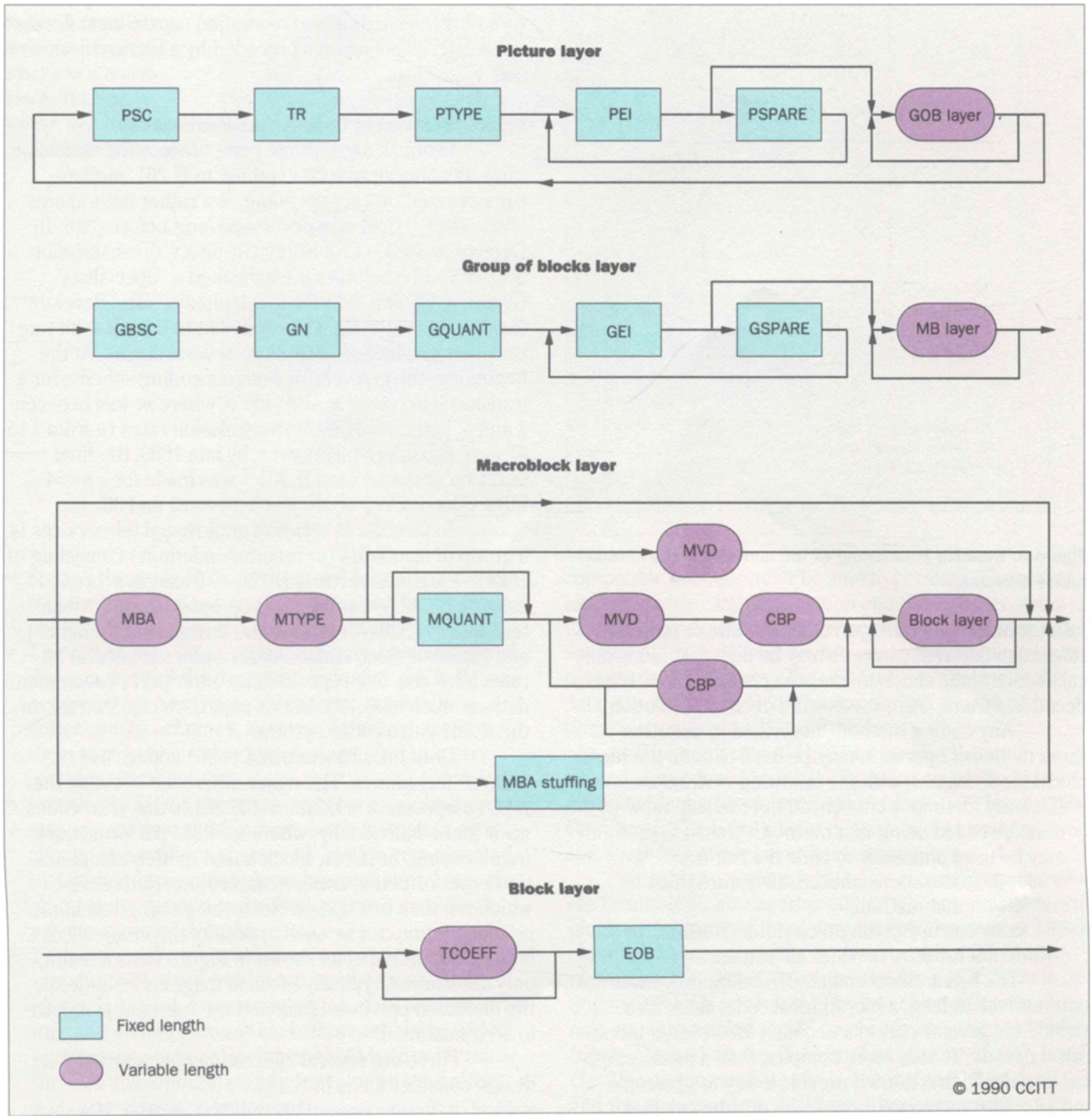


Figure 7. Syntax diagram of the video multiplex coder.[20]

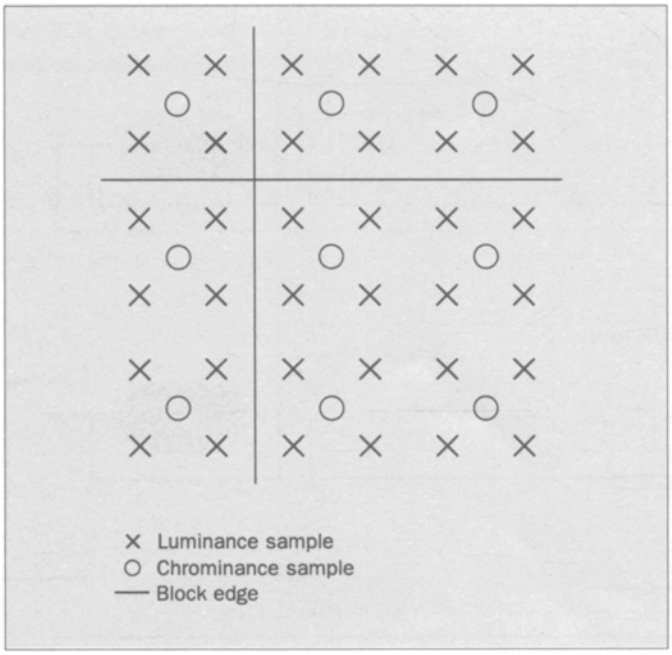


Figure 8. Relative positioning of the luminance and chrominance samples.

full-resolution input image. After that, one or more full-resolution difference images may be coded. A hierarchical decoder may abort the decoding process after it has decoded a frame that provides the desired resolution.

Any coding methods described in the other three modes of operation may be used to code the hierarchical mode frames, with the following restrictions:

- If a lossy method is chosen, all but the last frame must be coded using that method. A lossless method may be used optionally to code the last frame.
- If a lossless method is chosen, all frames must be coded with that method.
- The same entropy-coding technique (Huffman or arithmetic) must be used for all frames.

The hierarchical coding/decoding process is not symmetrical. Indeed, a hierarchical coder must also include the greater part of a decoder. However, a hierarchical decoder is only more complex than a nonhierarchical decoder in that it must provide a way to upsample and add. This increased complexity may be justified, given the flexibility afforded in matching the decoder to the application. This type of codec is well suited for "one-to-many" applications, as in a number of decoders (possibly having different resolution capabilities) accessing a database of images precoded by a hierarchical coder.

Videoconferencing Coding Standards H.261

From an algorithmic point of view, the extension from JPEG, intraframe DCT coding, to H.261, motion-compensated DCT video coding, is a rather natural one. Historically, H.261 was developed long before JPEG. In December 1984, CCITT Study Group XV (Transmission Systems and Equipment) established a "Specialists' Group on Coding for Visual Telephony." The development of this video transmission standard for low-bit-rate ISDN services has gone through several stages. At the beginning, the goal was to design a coding scheme for a transmission rate of $m \times 384$ kb/s, where m was between 1 and 5. Later, $n \times 64$ kb/s transmission rates (n from 1 to 5) were considered. However, by late 1989, the final CCITT recommendation H.261[20] was made for a $p \times 64$ kb/s video codec, where p is between 1 and 30.

In fact, the H series of audiovisual teleservices is a group of standards (or recommendations) consisting of H.221 — frame structure; H.230 — frame synchronous control; H.242 — communication between audiovisual terminals; H.320 — systems and terminal equipment; and H.261 — video codec. Audio codecs at several bit rates have also been specified by other CCITT recommendations, such as G.725. In this paper, we concentrate on the H.261 video codec system.

Both JPEG baseline and H.261 codecs use DCT and VLC techniques. The major difference between the JPEG compression scheme and H.261 is that JPEG codes each frame individually, whereas H.261 performs interframe coding. In H.261, block-based motion compensation is performed to compute interframe differences, which are then DCT coded. Here, the picture data in the previous frame can be used to predict the image blocks in the current frame, as shown in Figure 6. As a result, only differences, typically of small magnitude, between the displaced previous block and the current block have to be transmitted.

There are several interesting characteristics or design considerations in H.261.

- First, it defines essentially only the *decoder*. However, the *encoder*, which is not completely and explicitly specified by the standard, is expected to be compatible with the decoder.

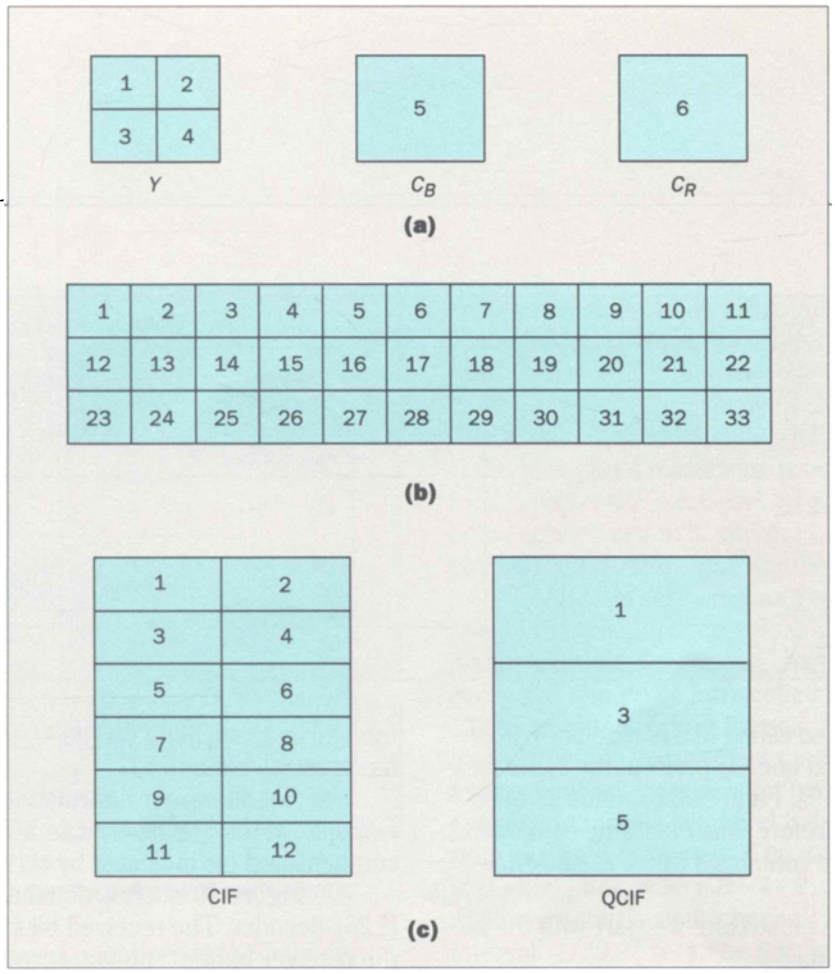


Figure 9. Successive arrangement of (a) blocks in a macroblock, (b) macroblocks in a GOB, and (c) GOBs in a picture.

- Second, because H.261 is designed for real-time communications, it uses only the closest previous frame as prediction to reduce the encoding delay.
- Third, it tries to balance the hardware complexities of the encoder and the decoder, since they are both necessary for a real-time videophone application. Other coding schemes, such as vector quantization (VQ), may have a rather simple decoder, but a very complex encoder.
- Fourth, H.261 is a compromise between coding performance, real-time requirement, implementation complexity, and system robustness. Motion-compensated DCT coding is a mature algorithm, and after years of study, quite general and robust in that it can handle various types of pictures.
- Fifth, the final coding structures and parameters are tuned more toward low-bit-rate applications. This choice is logical, because selection of the coding structure and coding parameters is more critical to codec performance at very low bit rates. At higher bit rates, the less-than-optimal parameter values do not affect codec performance very much.

Decoder Structures and Components. H.261 specifies a set of protocols that every compressed bit stream has to follow, and a set of operations that every standard compatible decoder must be able to perform. The actual hardware codec implementation and the encoder structure can vary drastically from one design to another. In a few places, user-defined bit streams may be inserted into the standard bit stream. We will first explain briefly the data structure in an H.261 bit stream and then the functional elements in an H.261 decoder.

The compressed H.261 bit stream[20] contains several layers (see Figure 7). They are *picture* layer, group of blocks (*GOB*) layer, macroblock (*MB*) layer, and *block* layer. The higher layer consists of its own header and a number of the lower layer data.

Only two picture formats — common intermediate format (CIF) and quarter-CIF (QCIF) — are allowed. CIF pictures are made of three components: luminance Y and color differences C_B and C_R, as defined in CCIR Recommendation 601. The CIF picture size for Y is 352 pixels per line by 288 lines per frame. The two-color difference signals are subsampled to 176 pixels per line and 144 lines per frame. Figure 8 shows the sampling pattern of Y, C_B, and C_R. The picture aspect ratio is 4(horizontal):3(vertical), and the picture rate is 29.97 non-interlaced frames per second. All standard codecs must be able to operate with QCIF; CIF is optional.

A picture frame is partitioned into 8 lines by 8

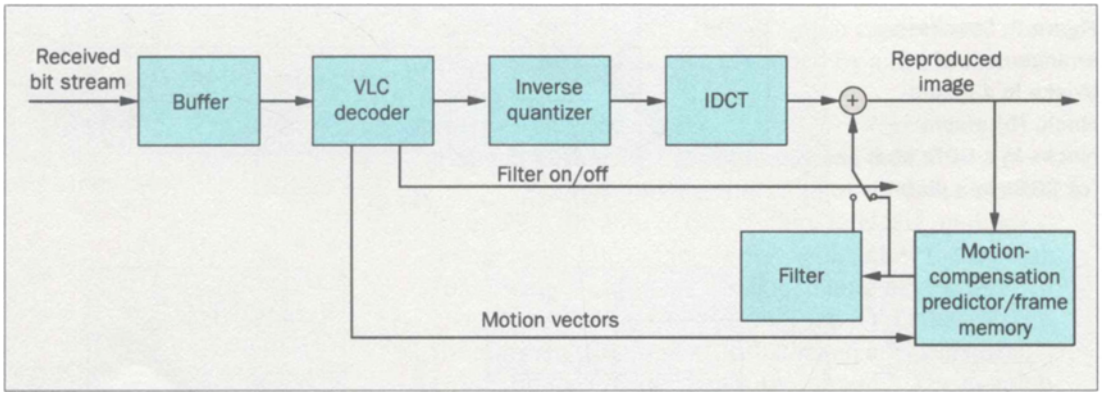


Figure 10. A typical H.261 decoder.

pixel image blocks. The so-called MB is made of 4 Y blocks, one C_B block, and one C_R block at the same location, as shown in Figure 9a. Figure 9b contains 33 MBs grouped into a GOB. Therefore, one CIF frame contains 12 GOBs and one QCIF frame contains 3 GOBs, as shown in Figure 9c.

In a compressed bit stream, we start with the picture layer. Its header contains:

- Picture start code (PSC) — a 20-bit pattern
- Temporal reference (TR) — a 5-bit input frame number
- Type information (PTYPE) — such as CIF/QCIF selection
- User-inserted bits.

Then, a number of GOB layer data follow.

At the GOB layer, a GOB header contains:

- Group of blocks start code (GBSC) — a 16-bit pattern
- Group number (GN) — a 4-bit GOB address
- Quantizer information (GQUANT) — quantizer step size normalized to lie in the range 1 to 31
- User-inserted bits.

Next come a number of MB layer data. An 11-bit stuffing pattern can be inserted repetitively right after a GOB header or after a transmitted macroblock.

At the MB layer, the header contains:

- Macroblock address (MBA) — VLC location relative to the previously coded MB
- Type information (MTYPE) — 10 types in total
- Quantizer (MQUANT) — normalized quantizer step size
- Motion vector data (MVD) — the differential displacement
- Coded block pattern (CBP) — the coded block location indicator.

The lowest layer is block layer, consisting of quantized transform coefficients (TCOEFF), followed by the end of block (EOB) symbol.

Not all header information need be present. For example, at the MB layer, if an MB is not motion-compensated (as indicated by MTYPE), MVD does not exist.

Figure 10 is a functional diagram of a typical H.261 decoder. The received bit stream is first kept in the receiver buffer. The VLC decoder decodes the compressed bits and distributes the decoded information to the elements that need that information. The VLC tables are given by the standard.

There are essentially four types of MBs:

- Intra — original pixels are transform-coded
- Inter — the difference pixels (with zero-motion vectors) are coded
- Inter with motion compensation (MC) — the displaced (nonzero-motion vectors) differences are coded
- Inter MC with filter — the displaced blocks are filtered by a predefined filter, which may help reduce visible coding artifacts at very low bit rates.

Certain MB types in this list allow the optional transmission of MQUANT and TCOEFF information. The received MTYPE information controls various switches at the decoder to produce the right combination.

A single-motion vector (horizontal and vertical displacement) is transmitted for one inter-MC macroblock, that is, the four Y blocks, one C_B, and one C_R block all share the same motion vector. The range of motion vectors is ±15 pixels with integer values. Using both MVD and MTYPE information, the predictor can choose the right pixels for prediction.

The transform coefficients of either the original or the differential pixels are ordered according to the zig-zag scanning pattern in Figure 11. These transform

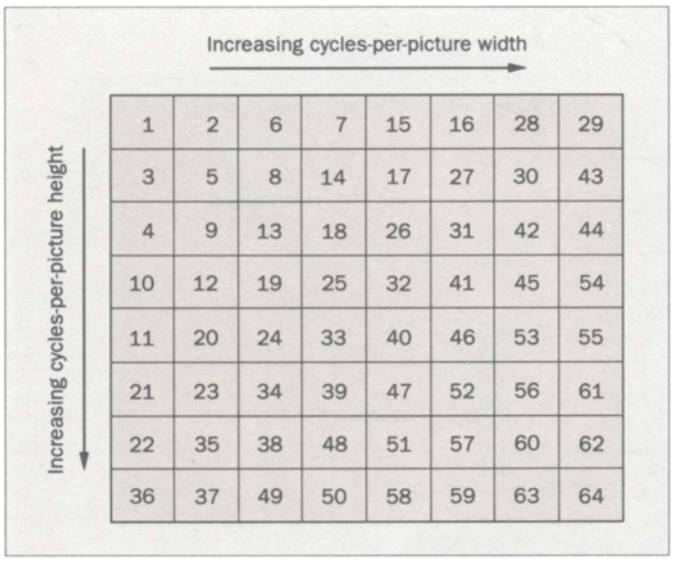


Figure 11. Transmission order for transform coefficients.

coefficients are selected and quantized at the encoder, and then variable-length-coded. Just as with JPEG, successive zeros between two nonzero coefficients are counted and called a *RUN*. The magnitude of a transmitted nonzero quantized coefficient is called a *LEVEL*. The most likely occurring combinations of (RUN, LEVEL) are encoded with the standard supplied VLC tables. The other combinations are coded with a 20-bit word consisting of a 6-bit ESCAPE code, 6 bits RUN, and 8 bits LEVEL. EOB is appended to the last nonzero coefficient, indicating the end of a block.

The inverse quantizer or the reconstruction process for all the coefficients other than the intra DC is defined by the following formula:

If *QUANT* is odd,

$$REC = QUANT \times (2 \times LEVEL + 1), \text{ for } LEVEL > 0,$$

$$REC = QUANT \times (2 \times LEVEL - 1), \text{ for } LEVEL < 0;$$

if *QUANT* is even,

$$REC = QUANT \times (2 \times LEVEL + 1) - 1, \text{ for } LEVEL > 0,$$

$$REC = QUANT \times (2 \times LEVEL - 1) + 1, \text{ for } LEVEL < 0,$$

where REC is the reconstructed value of a quantized coefficient. Almost all the reconstruction levels are odd numbers to reduce problems of mismatch between encoders and decoders from different manufacturers. The intra-DC coefficient is uniformly quantized with a fixed step size of 8, and coded with 8 bits.

The standard requires a compatible inverse DCT (IDCT) to be close to the ideal 64-bit floating point IDCT. H.261 specifies a measuring process for checking a valid IDCT. The peak error, mean error, and mean square error between the ideal IDCT and the IDCT under test have to be less than certain small numbers given in the standard.

A few other items are required by the standard. One of them is the image-block updating rate. To prevent mismatched IDCT error and channel error propagation, every MB should be intra-coded at least once in every 132 transmitted picture frames. The contents of the transmitted bit stream must meet the requirements of a *hypothetical reference decoder* (HRD). For CIF pictures, every coded frame is limited to fewer than 256 kb/s; for QCIF, the limit is 64 kb/s. The HRD receiving buffer size is $B + 256$ kb/s, where $B = 4 \times R_{max}/29.97$ and R_{max} is the maximum connection (channel) rate. At every picture interval (1/29.97 sec), the HRD buffer is examined. If at least one complete coded picture is in the buffer, then the earliest picture data are removed from the buffer and decoded. The buffer occupancy, right after the above data have been removed, must be less than B.

Encoder Constraints and Options. Figure 12 shows a typical encoder structure. For the purpose of this discussion, the elements inside a standard compatible encoder can be classified, based on their functionalities, into two categories:

- The *basic coding operation* units, such as motion estimator, quantizer, transform, and variable-word-length encoder (VLE)
- The *coding parameter decision* units, such as the coding control in Figure 12. These units select the parameter values of the basic operation units, including motion vectors, quantization step size, and picture frame rate.

Although H.261 does not explicitly specify a standard encoder, most basic operation elements are strongly constrained by the standard. However, other crucial elements, such as the parameter decision unit, are still open to the design engineers. We briefly outline our observations below.

The VLE implements the VLC H.261 tables. The forward DCT is not specified, but it is expected that the DCT inside the encoder matches the decoder IDCT, and the forward DCT should be able to match its own IDCT.

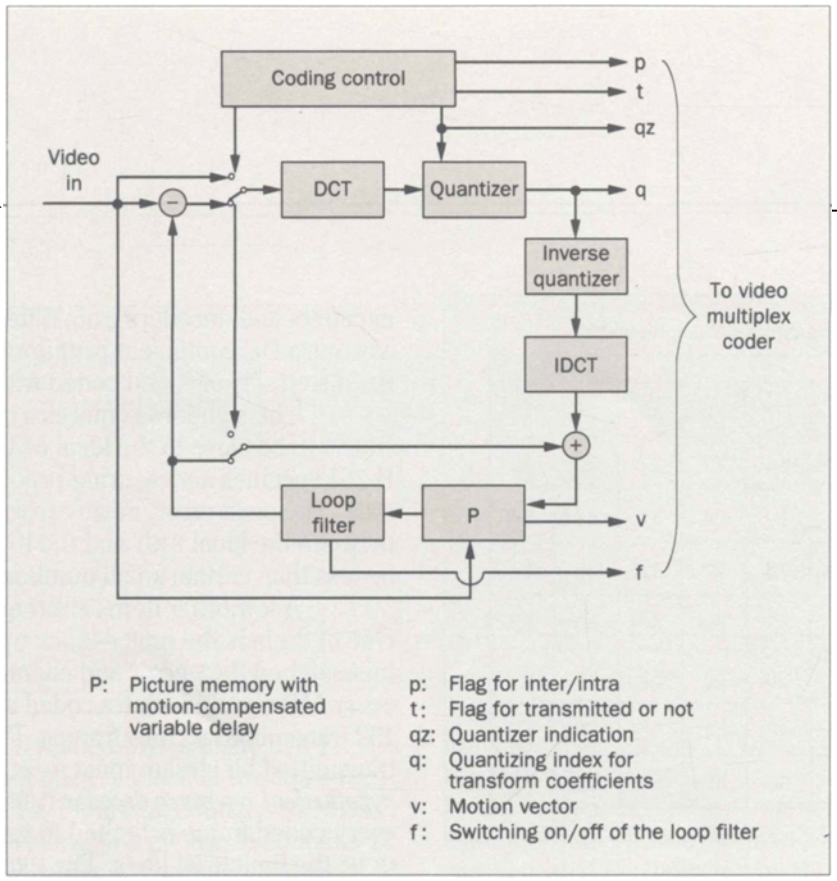


Figure 12. A typical H.261 encoder.

Because the inverse quantizer (IQ) is defined at the decoder, variations of the encoder quantizer are quite limited. From a theoretical viewpoint, however, it is not necessary for the decision levels of the encoder quantizer to be in the middle of two reconstruction levels. Also, encoder designers determine the criterion (a fixed or an adaptive threshold, for example) for selecting transform coefficients.

If motion compensation is selected, the motion estimator must be able to produce one motion vector for the entire MB. Block-matching motion estimation is used to produce such a motion vector; there can be several variations, such as hierarchical-motion estimation.[21] Because of the HRD model required by the standard, the encoded output bits of every frame must be regulated carefully. For example, successive frames producing small numbers of bits may violate the HRD requirement.

Although individual basic coding elements may affect the overall coding performance, the most critical and global influence on the encoder performance comes from the parameter decision units. The encoder must make several decisions, such as:

- How many frames should be transmitted, or conversely, how many should be skipped?
- What MTYPE should each macroblock use?
- What is the proper quantization step size?
- How do we control the buffer fullness so that it does not produce long delay and does not violate the HRD requirements?

Also, it is important to keep the hardware simple for practical applications. Many issues discussed have been investigated in the past; however, a complete solution has not been found.

MPEG First-Phase Standard

MPEG is an international standard[22–25] for the compression of digital audio and video transmission. The MPEG first-phase (MPEG-1) video compression standard, aimed primarily at coding video for digital storage media, at rates of 1 to 1.5 Mb/s, is well suited for a wide range of applications at a variety of bit rates. The standard mandates real-time decoding and supports features to facilitate interactivity with stored bit stream. It only specifies a syntax for the bit stream and the decoding process; sufficient flexibility is allowed for encoding complexity. Encoders can be designed for optimal tradeoff of performance versus complexity, depending on the specific application.

MPEG was chartered by the ISO to standardize a coded representation of video and audio suitable for digital storage media, such as compact disk – read-only memory (CD-ROM), digital audio tape (DAT), etc. The group's goal, however, has been to develop a *generic* standard, one that can be used in other digital video applications, such as telecommunication. The MPEG standard has three parts:

- Part 1 describes the synchronization and multiplexing of video and audio
- Part 2 describes video
- Part 3 describes audio.

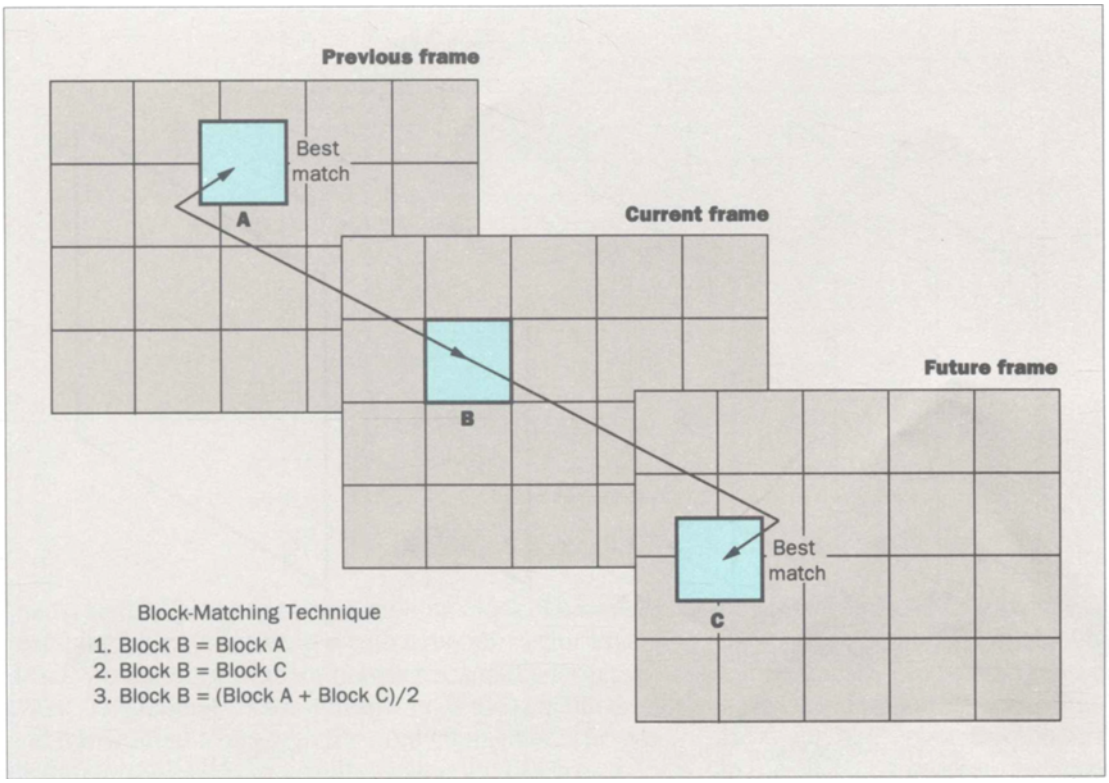


Figure 13. Motion-compensated interpolation.

An overview of the video portion of the MPEG standard follows.

Requirements of the Standard. Uncompressed digital video requires an extremely high transmission bandwidth. Digitized North American Television Standards Committee (NTSC) resolution video, for example, has a bit rate of approximately 100 Mb/s. With digital video, compression is necessary to reduce the bit rate to suit most applications. The required degree of compression is achieved by exploiting the spatial and temporal redundancy present in a video signal. However, the compression process is inherently lossy, and the signal reconstructed from the compressed bit stream is not identical to the input video signal. Compression typically introduces artifacts into the decoded signal.

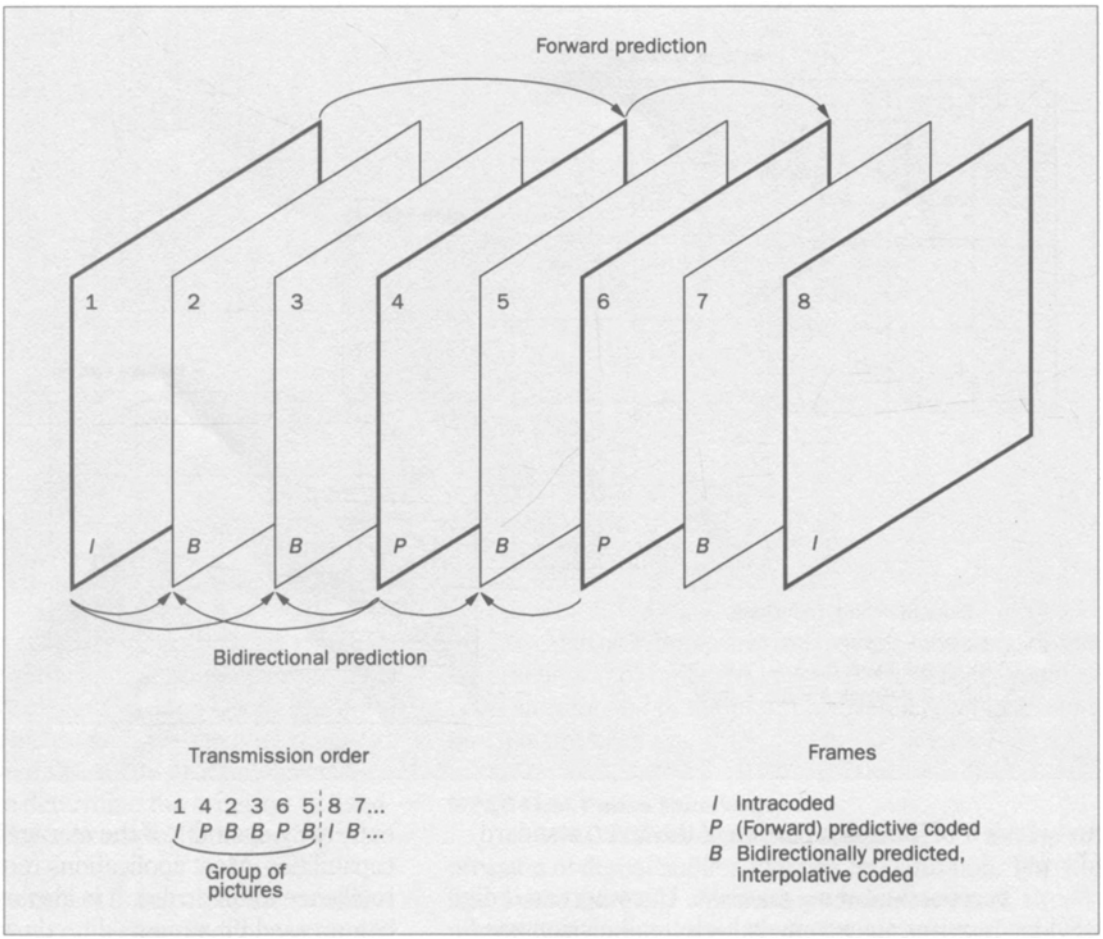
The primary requirement of the MPEG video standard is that it should achieve the highest possible quality of the decoded video at a given bit rate. In addition to picture quality, different applications stipulate additional requirements. For instance, multimedia applications require the ability to access, i.e., decode, any video frame in a short time. The ability to perform fast search directly on the bit stream — forward and backward — is extremely desirable if the storage medium has "seek" capabilities. Most applications require some degree of resilience to bit errors. It is also useful to be able to edit compressed bit streams directly while maintaining decodability. A variety of video formats should be supported.

Compression Algorithm Overview. References 23 and 25 describe the basic algorithms and syntax of the MPEG standard and Reference 25 details video coding using this standard. Here, we present the background and the basic information necessary for understanding this standard.

Exploiting spatial redundancy. The compression approach of MPEG video uses a combination of the ISO JPEG (still image) and CCITT H.261 (videoconferencing) standards. Because video is a sequence of still images, it is possible to compress or encode a video signal using techniques similar to JPEG. Such methods of compression are called intraframe coding techniques, where each frame of video is individually and independently compressed or encoded. Intraframe coding exploits the spatial redundancy that exists between adjacent pixels of a frame.

As in JPEG and H.261, the MPEG video-coding

Figure 14. Group of pictures.



algorithm employs a block-based two-dimensional DCT. A frame is first divided into 8×8 blocks of pixels, and the two-dimensional DCT is then applied independently on each block. This operation results in an 8×8 block of DCT coefficients in which most of the energy in the original (pixel) block is typically concentrated in a few low-frequency coefficients. A quantizer is applied to each DCT coefficient that sets many of them to zero. This quantization is responsible for the lossy nature of the compression algorithms in JPEG, H.261 and MPEG video. Compression is achieved by transmitting only the coefficients that survive the quantization operation and by entropy-coding their locations and amplitudes.

This standard allows the quantization operation to achieve a higher level of adaptation, a key factor in achieving good picture quality. Reference 26 details the relevant details of a quantizer adaptation scheme applicable within this context.

Exploiting temporal redundancy. Many of the interactive requirements discussed earlier can be satisfied by intraframe coding. However, as in H.261, the quality achieved by intraframe coding alone is not sufficient for typical video signals at bit rates around 1.5 Mb/s. Temporal redundancy results from a high degree of correlation between adjacent frames. The H.261 algorithm exploits this redundancy by computing a frame-to-frame difference signal called the *prediction error*. In computing the prediction error, the technique of motion compensation is employed to correct for motion. A block-based approach is adopted for motion compensation, where a block of pixels, called a *target block*, in the frame to be encoded is matched with a set of blocks of the same size in the previous frame, called a *reference frame*. The block in the reference frame that "best matches" the target block is used as the prediction for the latter, i.e., the prediction error is computed as the difference between

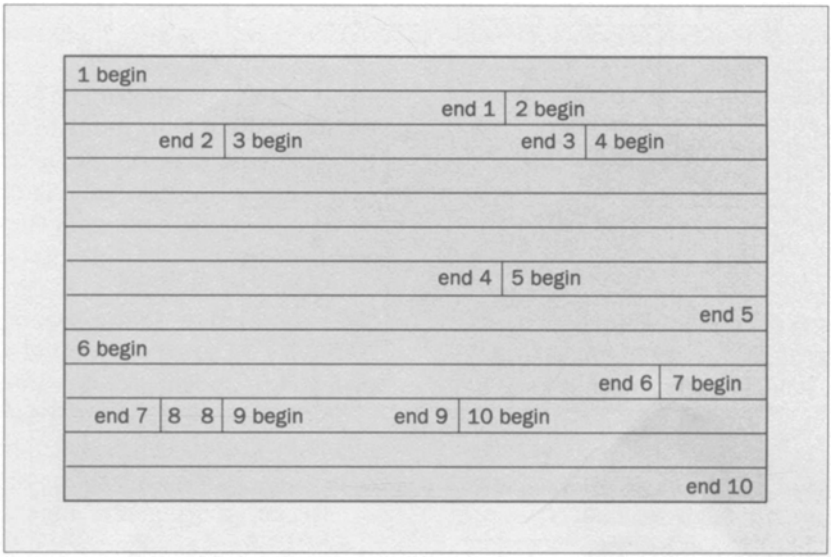


Figure 15. Possible arrangement of slices in a 256 X 192 picture.

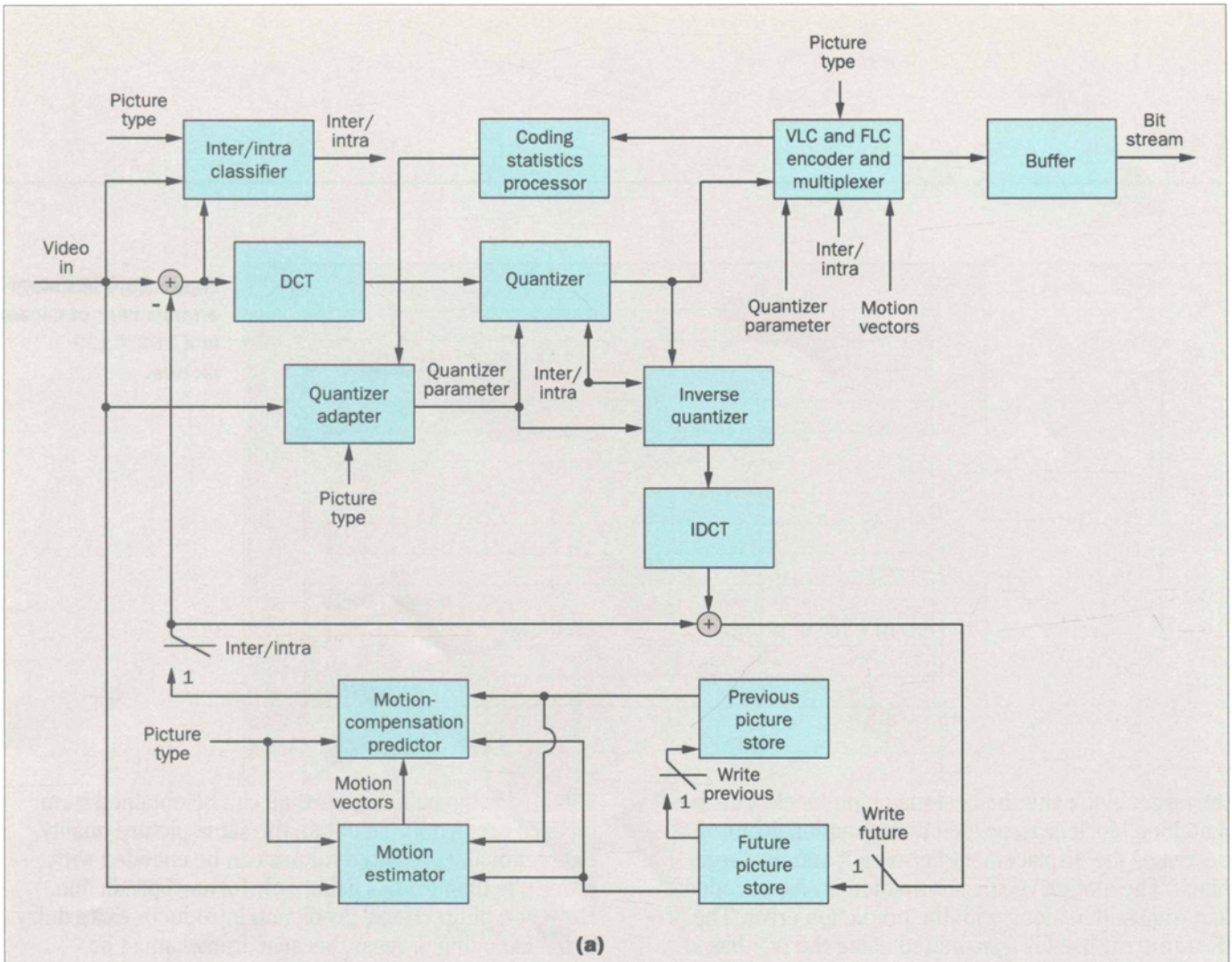
the target block and the best-matching block. This best-matching block is associated with a motion vector that describes the displacement between it and the target block. The motion vector information is also encoded and transmitted along with the prediction error. The prediction error itself is transmitted using the DCT-based intraframe encoding technique summarized above. In MPEG video (as in H.261), the block size for motion compensation is chosen to be 16×16, representing a reasonable tradeoff between the compression provided by motion compensation and the cost associated with transmitting the motion vectors.

Bidirectional temporal prediction. Bidirectional temporal prediction, also called motion-compensated interpolation, is a key feature of MPEG video. In bidirectional prediction, some of the video frames are encoded using two reference frames, one in the past and one in the future. A block in those frames can be predicted by another block from the past reference frame (*forward prediction*), or from the future reference frame (*backward prediction*), or by the average of two blocks, one from each reference frame (*interpolation*). In every case, the block from the reference frame is associated with a motion vector, so that two motion vectors are used with interpolation. Motion-compensated interpolation for a block in a bidirectionally predicted frame is illustrated in Figure 13. Frames that are bidirectionally predicted are never themselves used as reference frames.

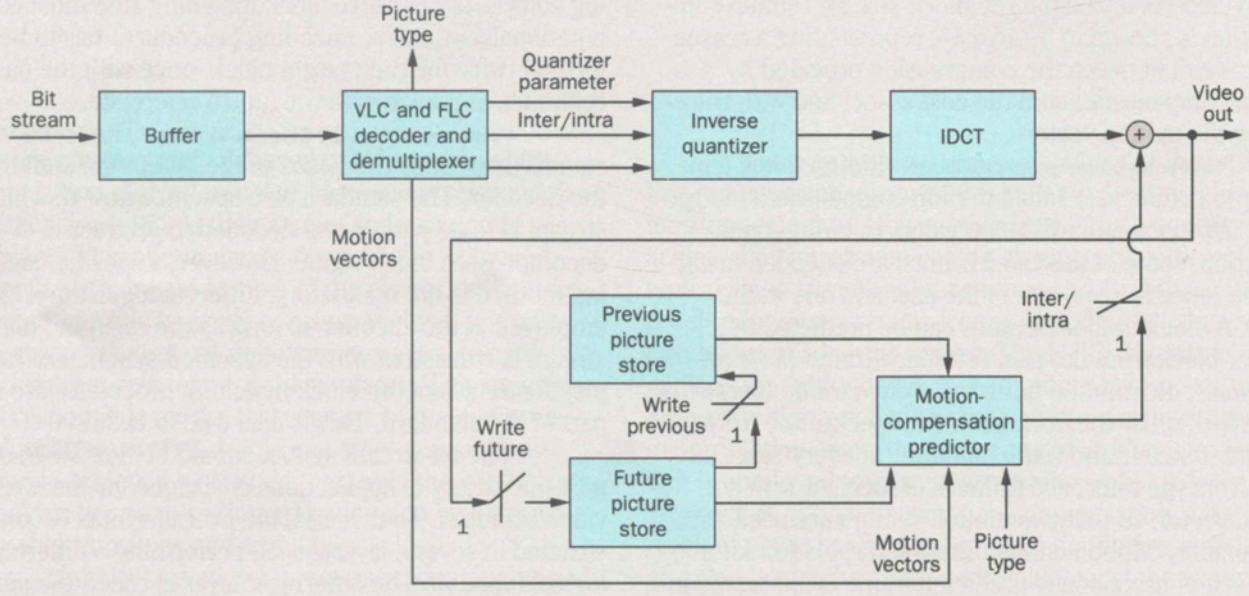
Bidirectional prediction provides a number of advantages. The primary one is that the compression obtained is typically higher than can be obtained from forward prediction. To obtain the same picture quality, bidirectionally predicted frames can be encoded with fewer bits than frames using only forward prediction. However, bidirectional prediction introduces extra delay in the encoding process, because frames must be encoded out of sequence. Further, it entails extra encoding complexity because block matching (the most computationally intensive encoding procedure) has to be performed twice for each target block, once with the past reference and once with the future reference.

Features of the Bit-Stream Syntax. The MPEG video standard specifies the *syntax* of the bit stream and, thus, the decoder. The standard also specifies how this bit stream is to be parsed and decoded to produce a decompressed video signal. However, a specific encoding method is not mandatory; different algorithms can be employed at the encoder so long as the resulting bit stream is consistent with the specified syntax. For example, the details of the block-matching procedure are not part of the standard. This is also true in H.261.

The bit-stream syntax should be flexible to support the variety of applications envisaged for the MPEG video standard. To this end, the overall syntax is constructed in several layers, each performing a different logical function. The outermost layer is called the *video sequence* layer, which contains basic parameters such as the size of the video frames, the frame rate, the bit rate, and certain other global parameters. A wide range of values is supported for all these parameters.



(a)



(b)

Inside the video sequence layer is the GOP layer, which provides support for random access, fast search, and editing. A sequence is divided into a series of GOPs, where each GOP contains an intracoded frame (I-frame) followed by an arrangement of (forward) predictive-coded frames (P-frames) and bidirectionally predicted, interpolative-coded frames (B-frames). Figure 14 shows a GOP example with six frames, 1 to 6. This GOP contains I-frame 1, P-frames 4 and 6, and B-frames 2, 3, and 5. The encoding and transmission order of the frames in this GOP is shown at the bottom of Figure 14. B-frames 2 and 3 are encoded after P-frame 4, using P-frame 4 and I-frame 1 as reference. We note that B-frame 7 in Figure 14 is part of the next GOP because it is encoded after I-frame 8. Random access and fast search are enabled by the availability of the I-frames, which can be decoded independently and serve as entry points for further decoding. The MPEG video standard allows GOPs to be of arbitrary structure and length. The GOP layer is the basic unit for editing an MPEG video bit stream.

The compressed bits produced by encoding a frame in a GOP constitute the *picture layer*. The picture layer first contains information on the type of frame that is present (I, P, or B), and the position of the frame in display order. The bits corresponding to the motion vectors and the DCT coefficients are packaged in the *slice* layer, the *macroblock* layer, and the *block* layer. Here, the block is the 8×8 DCT unit, the macroblock the 16×16 motion compensation unit, and the slice is a string of macroblocks of arbitrary length running from left to right and top to bottom across the frame. The slice layer is intended to be used for resynchronization during the decoding of a frame, in the event of bit errors. Prediction registers used in the differential encoding of motion vectors are reset at the start of a slice. It is again the responsibility of the encoder to choose the length of each slice. Figure 15 shows an example in which slice lengths vary throughout the frame. In the macroblock layer, the motion vector bits for a macroblock are followed by the block layer, which consists of the bits for the DCT coefficients of the 8×8 blocks in the macroblock. Figure 16 shows an MPEG video encoder and decoder. The different layers in the syntax and their use are illustrated in Table III.

Figure 16. A typical (a) MPEG-1 encoder and (b) MPEG-1 decoder.[25]

Table III. Layers of MPEG Video Bit-Stream Syntax

Syntax layer	Functionality
Sequence layer	Context unit
Group of pictures layer	Random access unit: video coding
Picture layer	Primary coding unit
Slice layer	Resynchronization unit
Macroblock layer	Motion compensation unit
Block layer	DCT unit

In demonstrations of MPEG video at a bit rate of 1.2 Mb/s, noninterlaced frames of size of 352 pixels by 240 lines at a frame rate of 29.97 per second have been used, with 2:1 color subsampling both horizontally and vertically. This resolution is roughly equivalent to one field of an interlaced NTSC frame. The quality achieved by the MPEG video encoder at this bit rate has often been compared to that of VHS. Although the MPEG video standard was originally intended for operation in the neighborhood of the above bit rate, a much wider range of resolution and bit rates is supported by the syntax. The MPEG video standard thus provides a generic bit-stream syntax that can be used for a variety of applications. The MPEG-video Committee Draft ISO CD 11172-2 provides all the details of the syntax, complete with informative sections on encoder procedures that are outside the scope of the standard.[22]

MPEG Second-Phase Standard. Currently, the second phase of MPEG (MPEG-2) is in progress. This phase is aimed at coding the video signals created by CCIR 601, e.g., 720 pixels, 480 lines, 30 frames per second, 2:1 interlace at bit rates of 2 Mb/s, or higher.

The first-phase standard, MPEG-1, focused on coding of single-layer (nonscalable) video of progressive format. The MPEG-2 standard is addressing issues of improved functionality by using scalable video coding. To initiate technical work for this phase of the MPEG standard, a worldwide video coding competition was held at Kurihama, Japan, in November 1991. Nearly 30 international organizations, including AT&T, submitted a video-coding scheme to this contest. AT&T's scheme was judged one of the best. Immediately after this competition, a collaborative phase of work began and, thus far, has resulted in a compromise scheme that retains many of the best features of the best performing schemes.

In the MPEG-2 standard, the main improvements in nonscalable coding result from emphasis on interlaced video. Various forms of frame/field motion-compensated

predictions have been adapted to increase the coding efficiency. Frame/field DCT coding and quantization have also been adapted. All optimization experiments are being performed at bit rates between 4 and 9 Mb/s.

The MPEG-2 standard is also addressing scalable video coding for a range of applications where video needs to be decoded and displayed at a variety of resolution scales. Among the noteworthy applications of interest are multipoint video conferencing, window display on workstations, video communications on asynchronous transfer mode networks, and high-definition television (HDTV) with embedded standard TV.

In scalable video coding, which can be achieved in the *spatial* or the *frequency* domain, it is assumed that given an encoded video bit stream, decoders of various complexities can decode and display appropriate-size replicas of the original video. A scalable video encoder and corresponding highest resolution decoder are likely to have increased complexity compared to a single-layer encoder/decoder. However, this increase in complexity may be well justified in applications where increased functionality and error resilience are important.

Conclusion

Image coding standards are crucial to the robust growth of visual services in communication and computer systems. Without them, communication between terminals and systems becomes extremely inconvenient and costly. In the absence of standards, economies of scale in the manufacture of user devices, board systems, and VLSI chips may be lost.

The JBIG, JPEG, P×64, and MPEG standards provide compression algorithms for all types of images that might be carried on multimedia services. With the onset of inexpensive chips, high-speed communication, and large capacity disk storage, all elements needed for rapid growth are in place.

References

1. J. R. Allen et al., "VCTV: A Video-on-Demand Market Test," *AT&T Technical Journal*, Vol. 72, No. 1, January/February 1993, pp. 7–14.
2. ISO Committee Draft 11544, *Coded representation of picture and audio information — Progressive bi-level image compression*, ISO/IEC IS 11544, to be published in 1993.
3. Horst Hampel et al., "Technical features of the JBIG standard for progressive bi-level image compression," *Signal Processing: Image Communication*, Vol. 4, No. 2, April 1992, pp. 103–110.
4. R. Hunter and A. H. Robinson, "International digital facsimile coding standards," *Proceedings of the IEEE*, Vol. 68, No. 7, July 1980, pp. 854–867.
5. CCITT Recommendation T.4, *Standardization of Group 3 facsimile apparatus for document transmission*, Geneva, 1980.
6. CCITT Recommendation T.6, *Facsimile coding schemes and coding control functions for Group 4 facsimile apparatus*, Malaga–Torremolinos, 1984.
7. ISO Committee Draft 10918-1, *Digital compression and coding of continuous-tone still images — Part 1: Requirements and guidelines*, ISO/IEC DIS 10918–1, 1991.
8. R. W. Hamming, *Coding and Information Theory*, Prentice-Hall, Englewood Cliffs, New Jersey, 1980, pp. 96–98.
9. A. S. Tanenbaum, *Computer Networks*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1981.
10. N. Abramson, *Information Theory and Coding*, McGraw-Hill, New York, N. Y., 1963, pp. 61–62.
11. *Digital Compression and Coding of Continuous-Tone Still Images, Part 2: Compliance Testing*, ISO/IEC CD 10918-2, 1991.
12. A. N. Netravali and B. G. Haskell, *Digital Pictures: Representation and Compression*, Plenum Press, New York, 1988.
13. D. A. Huffman, "A Method for the Construction of Minimum-Redundancy Codes," *Proc. IRE*, No. 40, September 1952, pp. 1098–1101.
14. J. Amsterdam, "Data Compression with Huffman Coding," *BYTE*, Vol. 11, No. 5, May 1986, pp. 99–108.
15. G. G. Langdon, Jr., "An Introduction to Arithmetic Coding," *IBM J. Res. Develop.*, Vol. 28, No. 2, March 1984, pp. 135–149.
16. I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic Coding for Data Compression," *Communications of the ACM*, Vol. 30, No. 6, June 1987, pp. 520–540.
17. N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete Cosine Transform," *IEEE Transactions on Computers*, Vol. C-23, No. 1, January 1974, pp. 90–93.
18. R. J. Clarke, *Transform Coding of Images*, Academic Press, Orlando, Florida, 1985.
19. H. Lohscheller, "A subjectively adapted image communication system," *IEEE Transactions on Communications*, Vol. COM-32, December 1984, pp. 1316–1322.
20. CCITT, *Recommendation H.261 — Video Codec for Audiovisual Services at px64 kbit/s*, Geneva, August 1990.
21. M. Bierling and R. Thoma, "Motion Compensating Field Interpolation Using a Hierarchically Structured Displacement Estimator," *Signal Processing*, Vol. 11, No. 4, Dec. 1986, pp. 387–404.
22. ISO Committee Draft 11172, *Information Technology-Coding of moving pictures and associated audio for digital storage media up to about 1.5 Mbit/s*, to be published in 1993.
23. D. J. LeGall, "MPEG: A Video Compression Standard for Multimedia Applications," *Communications of the ACM*, Vol. 34, No. 4, April 1991, pp. 47–58.
24. R. K. Jurgen, "Digital Video," *IEEE Spectrum*, Vol. 29, No. 3, March 1992, pp. 24–30.
25. A. Puri, "Video Coding Using the MPEG-1 Compression Standard," *Proc. International Symposium: Society for Information Display*, Boston, Massachusetts, May 1992, pp. 123–126.
26. A. Puri and R. Aravind, "Motion-Compensated Video Coding with Adaptive Perceptual Quantization," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. CSVT-1, December 1991, pp. 351–361.

(Manuscript received June 17, 1992)

Rangarajan Aravind *is a member of technical staff in the Visual Communications Research Department at AT&T Bell Laboratories in Holmdel, New Jersey. He works on algorithms for video compression, and is currently focusing on developing encoders for the Motion Picture Experts Group MPEG-1 and MPEG-2. Mr. Aravind joined AT&T in 1988, after receiving a B.S. from the Indian Institute of Technology, Madras, and an M.S. and Ph.D. from the University of California at Santa Barbara, all in electrical engineering.*

Glenn L. Cash *is a member of technical staff in the Visual Communications Research Department at AT&T Bell Laboratories in Holmdel, New Jersey. His research interests include image processing and coding. He is currently working on projects that use JPEG for still-image and motion video coding. Mr. Cash received a B.S.E.E. and M.S.E.E. from Monmouth College, West Long Branch, New Jersey. He joined AT&T in 1981.*

Donald L. Duttweiler *is a member of technical staff in the Visual Communications Research Department at AT&T Bell Laboratories in Holmdel, New Jersey. His research interests include adaptive filtering, speech coding, image coding, and very-large-scale-integration (VLSI) design. Mr. Duttweiler joined AT&T in 1970, after receiving a B.E.E. from Rensselaer Polytechnic Institute, Troy, New York, and an M.S. and Ph.D. from Stanford University, California, all in electrical engineering.*

Hsueh-Ming Hang *is currently associate professor in the Department of Electronics Engineering at National Chiao Tung University, Hsinchu, Taiwan. From 1984 to 1991, he was a member of technical staff at AT&T Bell Laboratories in Holmdel, New Jersey, where he conducted research in digital image compression algorithms and architecture. Mr. Hang received his B.S. and M.S. degrees from National Chiao Tung University, and a Ph.D. from Rensselaer Polytechnic Institute, Troy, New York.*

Barry G. Haskell *is head of the Visual Communications Research Department at AT&T Bell Laboratories in Holmdel, New Jersey. He is currently directing work in digital transmission and coding of images, videotelephone, satellite television transmission, and medical imaging. Mr. Haskell received a B.S., M.S., and Ph.D. in electrical engineering from the University of California, Berkeley. He joined AT&T in 1968.*

Atul Puri *is a member of technical staff in the Visual Communications Research Department at AT&T Bell Laboratories in Holmdel, New Jersey. He is working on image compression for videoconferencing, digital storage media, high-definition television, and packet networks. Mr. Puri received a B.S. from the Regional Engineering College, India, an M.S. from City College of New York, and a Ph.D. from the City University of New York, all in electrical engineering. Mr. Puri became a consultant to AT&T in 1985 while working on his Ph.D.; he joined AT&T in 1988.*