9

VIDEO PRESENTATION AND COMPRESSION

Borko Furht and Raymond Westwater
Florida Atlantic University
Department of Computer Science and Engineering
Boca Raton, Florida 33431
borko@cse.fau.edu

Abstract. This chapter we first presents various video representations and formats. Then, an overview of video compression techniques is given. Two standards and related techniques are described and evaluated – px64 Kbps (or H.261/H.263) standard for video-based communications and MPEG standard for intensive applications of full-motion video. Both standards use the combination of DCT-based intraframe compression and predictive interframe coding based on motion vector estimation. The techniques for motion vector estimation are analyzed as well. Finally, we also introduce the XYZ video compression technique based on 3D-DCT, which does not require motion vector estimation.

1. INTRODUCTION

This section first describes representation and formats for full-motion video, including computer and television formats. Then, we decompose a video sequence into information units, which are used for video compression and manipulation.

1.1 VIDEO REPRESENTATION AND FORMATS

Video represents a collection of images or frames, where each image (frame) can be represented in one of the formats discussed in the previous chapter. These formats include RGB, YUV, and YCrCb. Continuous motion is produced at a frame rate of 15 frames per second (fps) or higher. Full motion video is typically referred as one at 30 fps, while the traditional movies run at 24 fps. The NTSC television standard in U.S.A. uses 29.97 Hz frequency, which is approximately 30fps, while PAL and SECAM television standards use 25 fps. The new Showscan technology creates movies at 60 fps.

1.1.1 Computer Video Formats

Resolution of an image or video system refers to its capability to reproduce fine detail. Higher resolution requires more complex imaging and video systems for representing these images (video frames) in real time. In computer systems, resolution is characterized by the number of pixels. Table 1 summarizes popular computer video formats and related storage requirements.

Table 1. Characteristics of a Variety of Computer Video Formats

Computer Video Format	Resolution (pixels)	Colors (bits)	Storage Capacity Per Image
CGA - Color Graphics Adapter	320x200	4 (2 bits)	128,000 bits = 16 KB
EGA - Enhanced Graphics Adapter	640x350	16 (4 bits)	896,000 bits= 112 KB
VGA - Video Graphics Adapter	640x480	256 (8 bits)	2,457,600 bits= 307.2 KB
88514/A Display Adapter Mode	1024x768	256 (8 bits)	6,291,456 bits= 786.432 KB
XGA - Extended Graphics Array (a)	640x480	65,000 (24 bits)	6,291,456 bits= 786.432 KB
XGA - Extended Graphics Array (b)	1024x768	256 (8 bits)	6,291,456 bits= 786.432 KB
SVGA - Super VGA	1024x768	65,000 (24 bits)	2.36 MB

Computer video display system architecture typically consists of a frame memory and video controller, which is interfaced to the computer monitor.

1.1.2 Television Formats

In television systems, resolution refers to the number of line pairs resolved on the face of the display screen, expressed in cycles per picture height, or cycles per picture width. For example, the NTSC broadcast system in North America and Japan, referred as 525/59.94, has about 483 lines. The HDTV system doubles the number of lines of current broadcast television at approximately the same field rate. For example, a 1050x960 HDTV system has a total of 960 lines.

Spatial and temporal characteristics of conventional television systems (NTSC, SECAM, and PAL), and high-definition TV (HDTV) are presented in Tables 2 and 3, respectively.

Table 2. Spatial Characteristics of Television Systems

System	Total Lines	Active Lines	Vertical Resolution	Optimal Viewing Distance [m]	Aspect Ratio	Horizontal Resolution	Total Picture Elements
HDTV USA	1050	960	675	2.5	16/9	600	720,000
HDTV Europe	1250	1000	700	2.4	16/9	700	870,000
NTSC	525	484	242	7.0	4/3	330	106,000
PAL	625	575	290	6.0	4/3	425	165,000
SECAM	625	575	290	6.0	4/3	465	180,000

Aspect ratio in Table 2 specifies the geometry of the field and is defined as:

$$Aspect\_Ratio = \frac{Width}{Height}$$

Conventional television systems have the aspect ratio 4/3 = 1.33, while the aspect ratio of HDTV system is 16/9 = 1.78.

The viewing distance (D) determines the angle (h) subtended by the picture height (H), and is defined as:

$$h = \frac{D}{N}$$

Optimal viewing distance for NTSC systems is 7.0 meters, for PAL and SECAM, 6.0 meters, and for HDTV U.S.A. 2.5 meters.

Table 3. Temporal Characteristics of Television Systems

System	Total Channel Width [MHz]	Video Baseband Y [MHz]	Video Baseband R-Y [MHz]	Video Baseband B-Y [MHz]	Scanning Rate Camera [Hz]	Scanning Rate HDTV Display [Hz]	Scanning Rate Convent. Display [Hz]
HDTV U.S.A.	9.0	10.0	5.0	5.0	59.94	59.94	59.94
HDTV Europe	12.0	14.0	7.0	7.0	50	100	50
NTSC	6.0	4.2	1.0	0.6	59.94	NA	59.94
PAL	8.0	5.5	1.8	1.8	50	NA	50
SECAM	8.0	6.0	2.0	2.0	50	NA	50

Television video signals are transmitted through a single television channel. For example, NTSC channel requires 6 MHz frequency, and luminance, chrominance, and sound carriers are separated to avoid interference, as illustrated in Figure 1.

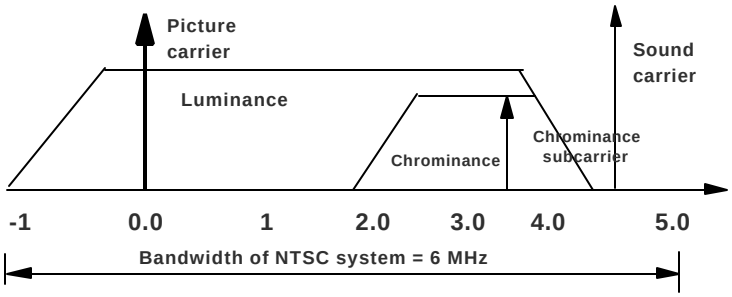

Figure 1. Transmission of TV video signals.

In television systems, combining individual components (RGB or YUV) can create a composite video signal into one signal. During the composition in one signal, chrominance signals can interfere with the luminance, and therefore appropriate modulation methods are used to eliminate this interference. In TV systems, RGB representation provides the highest quality (483 lines per frame in the case of NTSC). S-video includes two components, one chrominance and one luminance, and provides 400 lines per frame, while composite signal, which is a single line, gives 200 lines per frame. RF video signal gives the lowest quality.

1.2 VIDEO INFORMATION UNITS

When the motion video is represented in digital form, it can be decomposed into a time-dependent sequence of individual information units. For example, a motion video sequence can be divided into film, clips, frames, blocks, and pixels, as illustrated in Figure 2.

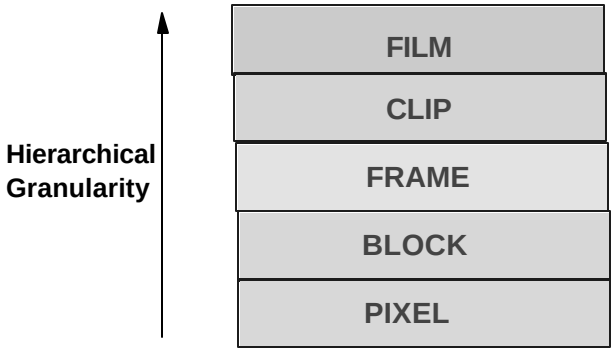

Figure 2. Motion video sequence divided into information units.

A full motion video, or film, consists of a number of clips, which are characterized with a common thread (for example, a camera shot). Each clip consists of a number of frames. Each frame can be divided into blocks. Typical sizes of the blocks, which are used in video processing systems (such as compression, retrieval and indexing, motion estimation, etc.) are 8x8 and 16x16 pixels. Pixels are the smallest pieces of information, which consist of 8, 16, or 24 bits.

2. OVERVIEW OF VIDEO COMPRESSION TECHNIQUES

Most digital images contain a high degree of redundancy, which means that an efficient compression technique can significantly reduce the amount of information needed to store or transmit them. This redundancy can be found between single pixels, between lines, or between frames, when a scene is stationary or slightly moving.

Since about 1989, moving digital video images have been integrated with programs. The difficulty in implementing moving digital video is the tremendous bandwidth required for the encoding of video data. For example, a quarter screen image (320 x 240 pixels) playing on an RGB video screen at full speed of 30 fps requires storage and transmission of 6.9 million bytes per second (MB/s). This data rate is simply prohibitive, and so means of compressing digital video suitable for real-time playback are a necessary step for the widespread introduction of digital motion video applications.

Many digital video compression algorithms have been developed and implemented. The compression ratios of these algorithms vary according to the subjective acceptable level of error, the definition of the word compression, and who is making the claim. Table 4 summarizes video compression algorithms, their typical compression ratios reported in the literature, and their characteristics.

Table 4. Overview of Video Compression Algorithms

Compression Algorithm	Typical Compression Ratio	Characteristics
Intel RTV/Indeo	**3:1**	A 128X240 data stream is interpolated to 256X240. Color is subsampled 4:1. A simple 16 bit codebook is used without error correction. Frame differencing is used.
Intel PLV	**12:1**	A native 256X240 stream is encoded using vector quantization and motion compensation. Compression requires specialized equipment.
IBM Photomotion	**3:1**	An optimal 8-bit color palette is determined, and run-length encoding and frame differencing are used.
Motion JPEG	**10:1**	Uses 2-D DCT to encode individual frames. Gives good real-time results with inexpensive but special-purpose equipment. This technique supports random-access since no frame differencing is used.
Fractals	**10:1**	Fractals compress natural scenes well, but require tremendous computing power.
Wavelets	**20:1**	2-D and 3-D wavelets have been used in the compression of motion video. Wavelet compression is low enough in complexity to compress entire images, and therefore does not suffer from the boundary artifacts seen in DCT-based techniques.
H.261/H263	**50:1**	Real-time compression and decompression algorithm for video telecommunications. It is based on 2-D DCT with simple motion estimation between frames.
MPEG	**30:1**	Uses 2-D DCT with motion estimation and interpolation between frames. The MPEG standard is difficult and expensive to compress, but plays back in real time with inexpensive equipment.

An ideal video compression technique should have the following characteristics:

- Will produce levels of compression rivaling MPEG without objectionable artifacts.
- Can be played back in real time with inexpensive hardware support.
- Can degrade easily under network overload or on a slow platform.
- Can be compressed in real time with inexpensive hardware support.

In this chapter, we focus on two well-established standard techniques, MPEG for intensive video applications and H.261/H.263 standard for video telecommunications. We also describe the promising XYZ technique.

Video compression techniques made feasible a number of applications. Four distinct applications of the compressed video can be summarized as: (a) consumer broadcast television, (b) consumer playback, (c) desktop video, and (d) videoconferencing.

Consumer broadcast television, which includes digital video delivery to homes, typically requires a small number of high-quality compressors and a large number of low-cost decompressors. Expected compression ratio is about 50:1.

Consumer playback applications, such as CD-ROM libraries and interactive games, also require a small number of compressors and a large number of low-cost decompressors. The required compression ratio is about 100:1.

Desktop video, which includes systems for authoring and editing video presentations, is a symmetrical application requiring the same number of encoders and decoders. The expected compression ratio is in the range from 5:1 to 50:1.

Videoconferencing applications also require the same number of encoders and decoders, and the expected compression ratio is about 100:1.

Table 5. Applications of the Compressed Video and Current Video Compression Standards

Application	Bandwidth	Standard	Size	Frame Rate [frames/sec]
Analog Videophone	5-10 Kbps	none	170x128	2-5
Low Bit-rate Video Conferencing	26-64 Kbps	H.263	128x96 176x144	15-30
Basic Video Telephony	64-128 Kbps	H.261	176x144 352x288	10-20
Video-conferencing	>= 384 Kbps	H.261	352x288	15-30
Interactive Multimedia	1-2 Mbps	MPEG-1	352x240	15-30
Digital TV - NTSC	3-10 Mbps	MPEG-2	720x480	30
High Definition Television	15-80 Mbps	MPEG-2	1200x800	30-60

Table 5 summarizes applications of the compressed video, by specifying current standards used in various applications, the required bandwidth, and the typical frame sizes and frame rates.

3. Px64 COMPRESSION ALGORITHM FOR VIDEO TELECOMMUNICATIONS

The H.261/263 standard, commonly called px64 Kbps, is optimized to achieve very high compression ratios for full-color, real-time motion video transmission. The px64 compression algorithm combines intraframe and interframe coding to provide fast processing for on-the-fly video compression and decompression. *Intraframe coding* refers to the coding of individual frames, while *interframe coding* is the coding of a frame in reference to the previous or future frames.

The px64 standard is optimized for applications such as video-based telecommunications. Because these applications are usually not motion-intensive, the algorithm uses limited motion search and estimation strategies to achieve higher compression ratios. For standard video communication images, compression ratios of 100:1 to over 2000:1 can be achieved.

The px64 compression standard is intended to cover the entire ISDN channel capacity (p = 1, 2,…30). For p = 1 to 2, due to limited available bandwidth, only desktop face-to-face visual communications (videophone) can be implemented using this compression algorithm. However, for p >6, more complex pictures are transmitted, and the algorithm is suitable for videoconferencing applications.

3.1 PICTURE FORMATS FOR H.261/H.263 VIDEO CODECS

For the H.261 algorithm two formats are defined: CIF and QCIF, while for the H.263 algorithm three additional formats are specified: SQCIF, 4CIF, and 16CIF.

The Common Intermediate Format (CIF) is a non-interlaced format, based on 352x288 pixels per frame at 30 frames per second. These values represent half the active lines of 625/25 television signal and the picture rate of a 525/30 NTSC signal. Therefore, 625/25 systems need only to perform a picture rate conversion, while NTSC systems need to perform only a line-number conversion.

Color pictures are coded using one luminance and two color-difference components (YCbCr format), specified by the CCIR 601 standard. The Cb and Cr components are subsampled by a factor of two on both horizontal and vertical directions, and have 176x144 pixels per frame. The picture aspect ratio for all five CIF-based formats is 4:3. Table 6 summarizes the picture formats for H.261 and H.263 codecs.

Table 6. Picture Formats for H.261 and H.263 Video Codecs

PICTURE FORMAT	LUMINANCE PIXELS	MAX FRAME RATE [F/S]	VIDEO SOURCE RATE	AVERAGE CODED BIT RATE	H.261 CODEC	H.263 CODEC
SQCIF	128 x 96	30	1.3 Mb/s	26 Kb/s	Optional	Required
QCIF	176 x 144	30	9 Mb/s	64 Kb/s (px64 Kbps)	Required	Required
CIF	352 x 288	30	36 Mb/s	384 Kb/s (px64 Kbps)	Optional	Optional
4CIF	704 x 576	30	438 Mb/s	3-6 Mb/s	Not defined	Optional
16CIF	1408 x 1152	50	2.9 Gb/s	20-60 Mb/s	Not defined	Optional

Example 1: Desktop videophone application

For a desktop videophone application, if we assume p = 1, the available ISDN network bandwidth is B_A = 64 Kbits/s. If the QCIF format is used, the required number of bits per frame consists of one luminance and two chrominance components:

$$N_b = (144 \times 176 + 72 \times 88) \times 8\ bits = 300\ Kbits/\ frame$$

If the data is transmitted at 10 frames/s, the required bandwidth is:

$$B_r = 300 Kbits/\ frame \times 10\ frames/s = 3 bits/s$$

As a consequence, a video compression algorithm should provide compression ratio of minimum:

$$C_r = \frac{B_r}{B_A} = \frac{3\ Mbits/s}{64\ Kbits/s} = 47$$

Example 2: Videoconferencing application

If we assume p = 10 for a videoconferencing application, the available ISDN network bandwidth becomes B_r = *640* Kbits/s. If the CIF format is used, the total number of bits per frame becomes:

$$N_b = (288 \times 352 + 144 \times 176 + 144 \times 176) \times 8\ bits = 1.21\ bits/\ frame$$

Assuming a frame rate of 30 frames/s, the required bandwidth for the transmission of videoconferencing data becomes:

$$B_r = 1.21\ Mbits/\ frame \times 30\ frames/s = 36.4\ Mbits/s$$

Therefore, a video compression algorithm should provide compression ratio of minimum:

$$C_r = \frac{B_r}{B_a} = \frac{36.4 Mbits/s}{640 Kbits/s} = 57$$

3.2 THE H.261/H.263 VIDEO ENCODER

The H.261/H.263 video compression algorithm combines intraframe and interframe coding to provide fast processing combines intraframe and interframe coding to provide fast processing for on-the-fly video. The algorithm creates two types of frames:

- DCT-based intraframe compression, which similar to JPEG, uses DCT, quantization, and entropy coding, and
- Predictive interframe coding based on Differential Pulse Code Modulation (DPCM) and motion estimation.

The block diagram of the video encoder is presented in Figure 3.

The H.261/H.263 coding algorithm begins by coding an intraframe block and then sends it to the video multiplex coder. The same frame is then decompressed using the inverse quantizer and inverse DCT, and then stored in the frame memory for interframe coding.

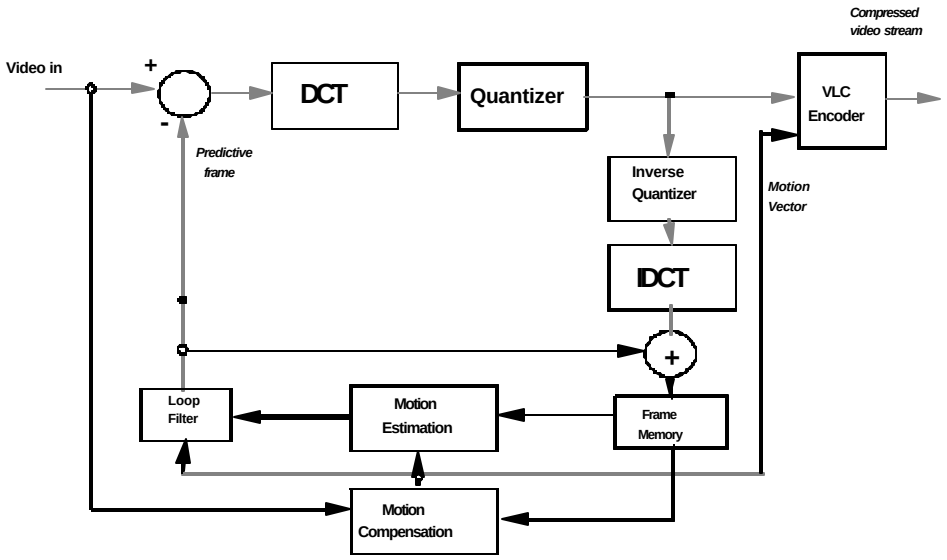

Figure 3. Block diagram of the H.261/H.263 video encoder.

During the interframe coding, the prediction based on the DPCM algorithm is used to compare every macro block of the actual frame with the available macro blocks of the previous frame, as illustrated in Figure 4. To reduce the encoding delay, only the closest previous frame is used for prediction. Then, the difference, created as error terms, is DCT-coded and quantized, and sent to the video multiplex coder with or without the motion vector.

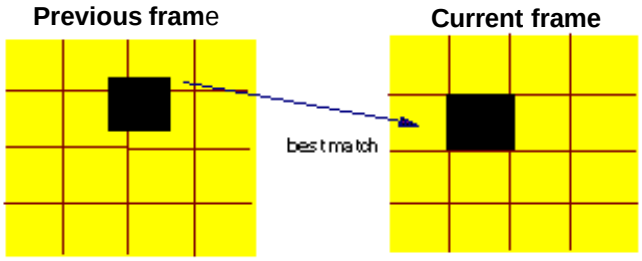

Figure 4. The principle of interframe coding in H.261/263 codecs.

For interframe coding, the frames are encoded using one of the following three techniques:

(i) DPCM coding with no motion compensation (zero-motion vectors).
(ii) DPCM coding with non-zero vectors.
(iii) Blocks are filtered by an optional predefined filter to remove high-frequency noise.

At the final step, variable-length coding (VLC), such as Huffman encoder, is used to produce more compact code. An optional loop filter can be used to minimize the prediction error by smoothing the pixels in the previous frame. At least one in every 132 frames should be an intraframe.

3.3 THE H.261/H.263 VIDEO DECODER

The H.261/H.263 video decoder is shown in Figure 5. It consists of the receiver buffer, VLC decoder, inverse quantizer, inverse DCT, and the motion compensation, which includes frame memory and an optional loop filter [3,4].

In addition to the encoding and decoding of video, the audio data must also be compressed and decompressed. Special buffering and multiplexing/demultiplexing circuitry is required to handle the complexities of combining the video and audio.

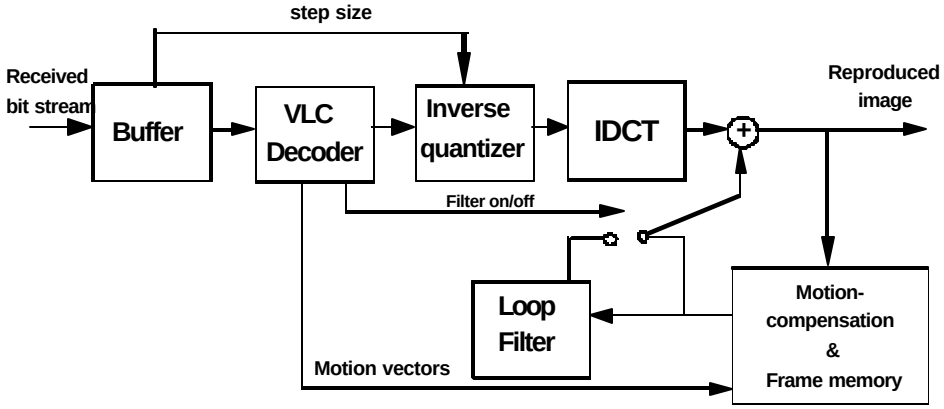

Figure 5. Block diagram of the H.261/H.263 video decoder.

3.4 VIDEO DATA STRUCTURE

According to the H.261 standard, a data stream has a hierarchical structure consisting of Pictures, Groups of Blocks (GOB), Macro Blocks (MB), and Blocks [3,4]. A Macro Block is composed of four (8 x 8) luminance (Y) blocks, and two (8 x 8) chrominance (C_r and C_b) blocks, as illustrated in Figure 6.

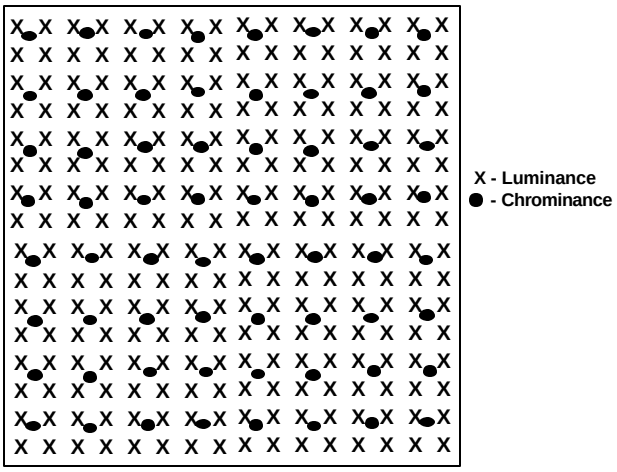

Figure 6. The composition of a Micro Block: MB = 4Y + Cb + Cr.

A Group of Blocks is composed of 3x11 MBs. A CIF Picture contains 12 GOBs, while a QCIF Picture consists of four GOBs. The hierarchical block structure is shown in Figure 7.

Each of the layers contains headers, which carry information about the data that follows. For example, a picture header includes a 20-bit picture start code, video format (CIF or QCIF), frame number, etc. A detailed structure of the headers is given by Liou [3].

A H.261 codec, proposed by Ghanbari [5], expands the existing H.261 codec to operate in ATM networks. A software-based video compression algorithm, called the Popular Video Codec (PVC), proposed by Huang et al. [6], is suitable for real-time systems. The PVC coder simplifies compression and decompression processes of the px64 algorithm by removing the transform and the motion estimation parts, and modifies the quantizer and the entropy coder.

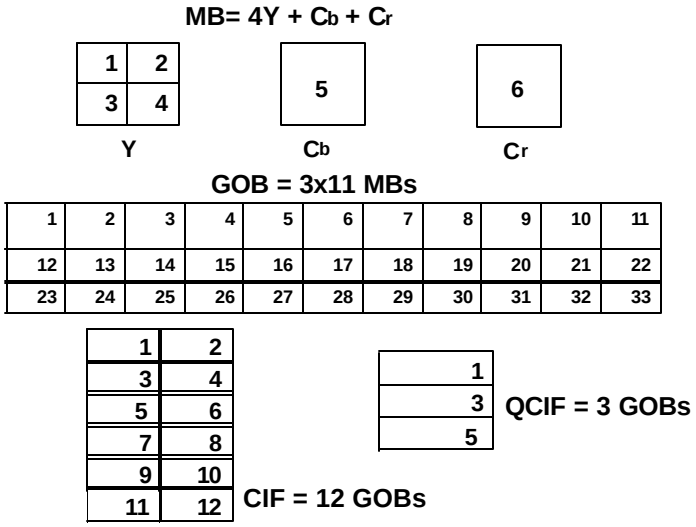

Figure 7. Hierarchical block structure of the p x 64 data stream.

4. THE MPEG VIDEO COMPRESSION STANDARD

The MPEG compression algorithm is intended for compression of full-motion video [8,10]. The compression method uses interframe compression and can achieve compression ratios of 200:1 through storing only the differences between successive frames. The MPEG approach is optimized for motion-intensive video applications, and its specification also includes an algorithm for the compression of audio data at ratios ranging from 5:1 to 10:1.

The MPEG first-phase standard (MPEG-1) is targeted for compression of 320x240 full-motion video at rates of 1 to 1.5 Mb/s in applications, such as interactive multimedia and broadcast television. MPEG-2 standard is intended for higher resolutions, similar to the digital video studio standard, CCIR 601, EDTV, and further leading to HDTV. It specifies compressed bit streams for high-quality digital video at the rate of 2-80 Mb/s. The MPEG-2 standard supports interlaced video formats and a number of features for HDTV.

The MPEG-2 standard also addresses scalable video coding for a variety of applications, which need different image resolutions, such as video communications over ISDN networks using ATM [7,8]. The MPEG-4 standard is intended for compression of full-motion video consisting of small frames and requiring slow refreshments. The data rate required is 9-40 Kbps, and the target applications include interactive multimedia and video telephony. This standard requires the development of new model-based image coding techniques for human interaction and low-bit-rate speech coding techniques [7]. Table 7 illustrates various motion-video formats and corresponding MPEG parameters.

The MPEG algorithm is intended for both asymmetric and symmetric applications. Asymmetric applications are characterized by frequent use of the decompression process, while the compression process is performed once. Examples include movies-on-demand, electronic publishing, and education and training. Symmetric applications require equal use of the compression and decompression processes. Examples include multimedia mail and video-conferencing.

Table 7. Parameters of MPEG Algorithms

FORMAT	VIDEO PARAMETERS	COMPRESSED BIT RATE	
SIF	352 x 240 at 30 Hz	1.2-3 Mbps	→ MPEG-1
CCIR 601	720 x 486 at 30 Hz	5-10 Mbps	
EDTV	960 x 486 at 30 Hz	7-15 Mbps	→ MPEG-2
HDTV	1920 x 1080 at 30 Hz	20-40 Mbps	

When the MPEG standard was conceived, the following features were identified as being import: random access, fast forward/reverse searches, reverse playback, audio-visual synchronization, robustness to errors, editability, format flexibility, and cost-trade-off. These features were described in detail by LeGall [8].

The MPEG standard consists of three parts: synchronization and multiplexing of video and audio; video; and audio.

4.1 MPEG FRAME STRUCTURE

In the MPEG standard, frames in a sequence are coded using three different algorithms, as illustrated in Figure 8.

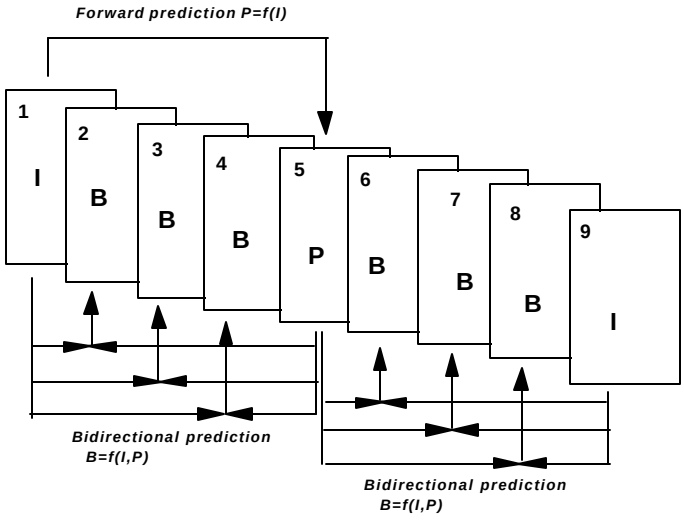

Figure 8. Types of frames in the MPEG standard.

I frames (intra images) are self-contained and coded using a DCT-based technique similar to JPEG. **I** frames are used as random access points to MPEG streams, and they give the lowest compression ratios within MPEG.

P frames (predicted images) are coded using forward predictive coding, where the actual frame is coded with reference to a previous frame (**I** or **P**). This process is similar to H.261 predictive coding, except the previous frame is not always the closest previous, as in H.261 coding (see Figure 4). The compression ratio of **P** frames is significantly higher than of **I** frames.

B frames (bidirectional or interpolated images) are coded using two reference frames, a past and a future frame (which can be **I** or **P** frames). Bidirectional or interpolated coding provides the highest amount of compression.

Note that in Figure 8, the first three **B** frames (2, 3 and 4) are bidirectionally coded using the past frame **I** (frame 1) and the future frame **P** (frame 5). Therefore, the decoding order will differ from the encoding order. The **P** frame 5 must be decoded before **B** frames 2, 3, and 4, and **I** frame 9 before **B** Frames 6, 7, and 8. If the MPEG sequence is transmitted over the network, the actual transmission order should be {1, 5, 2, 3, 4, 9, 6, 7, 8}.

The MPEG application determines a sequence of **I**, **P**, and **B** frames. If there is a need for fast random access, the best resolution would be achieved by coding the whole sequence as **I** frames (MPEG become identical to MJPEG). However, the highest compression ratio can be achieved by incorporating a large number of **B** frames. The following sequence has been proven to be very effective for a number of practical applications [7]:

(I B B P B B P B B) (I B B P B B P B B)…

In the case of 25 frames/s, random access will be provided through nine still frames (**I** and **P** frames), which is about 360 ms [7]. On the other hand, this sequence will allow a relatively high compression ratio. In an example, if we assume that the compression ratio for **I** frames is 1:10, for **P** frames is 1:40, and for **B** frames is 1:90, an average compression ratio for this MPEG sequence is

$$Cr = \frac{1\times 10}{9} + \frac{2\times 40}{9} + \frac{6\times 90}{9} = 70$$

4.2 MPEG VIDEO ENCODER AND DECODER

The block diagram of the MPEG encoder is given in Figure 9, while the MPEG decoder is shown in Figure 10.

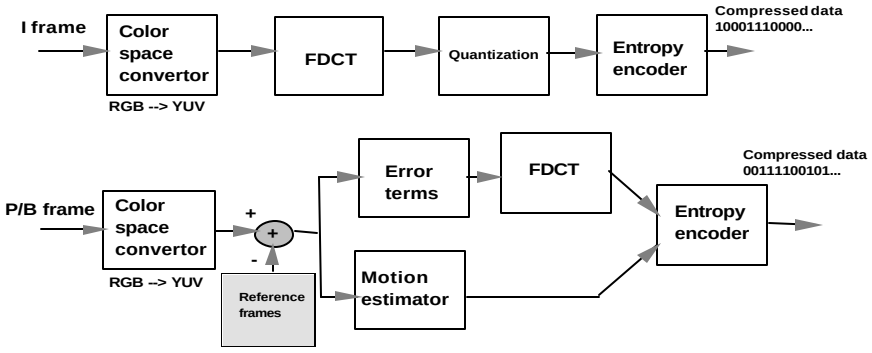

Figure 9. The block diagram of the MPEG encoder.

I frames are created similar to JPEG encoded pictures, while **P** and **B** frames are encoded in terms of previous and future frames. The motion vector is estimated, and the difference between the predicted and actual blocks (error terms) are calculated. The error terms are then DCT encoded and the entropy encoder is used to produce the compact code.

4.3 MPEG DATA STREAM

The MPEG specification defines a "video sequence" composed of a video sequence header and many Group-Of-Pictures (GOP), as illustrated in Figure 11. The video sequence header defines the video format, picture dimensions, aspect ratio, frame rate, and delivered data rate. Supported video formats include CCIR601, HDTV(16:9), and VGA. Supported chroma formats include "4:2:0" (YUV) and "4:4:4" (RGB). A suggested buffer size for the video sequence is also specified, a number intended to buffer jitter caused by differences in decode time.

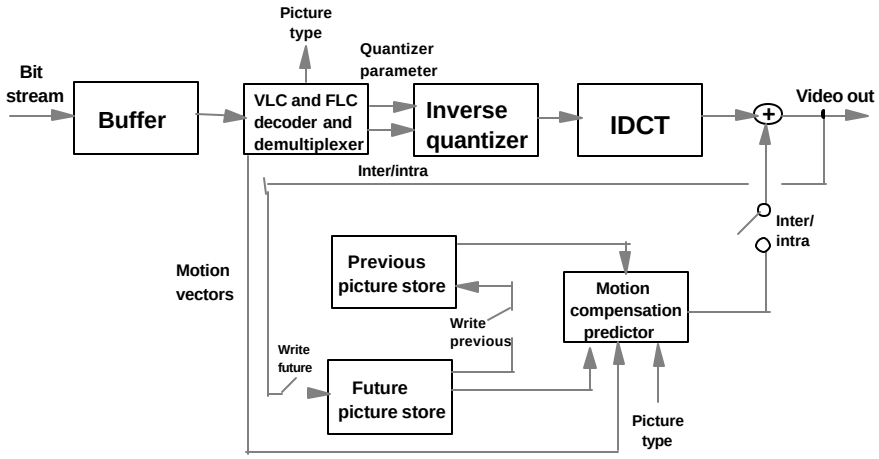

Figure 10. The block diagram of the MPEG decoder.

A GOP contains pictures that may be encoded into one of three supported compression formats. The GOP header contains a starting time for the group, and can therefore be used as a point of random access. Each frame within the GOP is numbered, and its number coupled with the GOP start time and the playback frame rate determines its playback time. Each picture is subdivided into "slices" and then into "macroblocks." A macroblock is composed of four 8x8 blocks of luminance data, and typically two 8x8 blocks of chrominance data, one Cr and one Cb.

I Picture Format

The **I** (Intraframe) picture format substantially corresponds to the JPEG format. These pictures are encoded by transformation into DCT space, quantization of the resultant coefficients, and entropy coding of the result. Transformation into DCT space is performed by an 8x8 DCT. Quantization is performed by reference to a user-loadable quantization table modified by a scale factor. This mechanism supports adaptive quantization at the cost of additional complexity – although 30% improvement in compression is claimed [9].

After quantization, the resulting coefficients are reordered in zig-zag order, run-length coded, variable-length coded, and entropy coded. The resulting data stream should roughly show JPEG levels of compression.

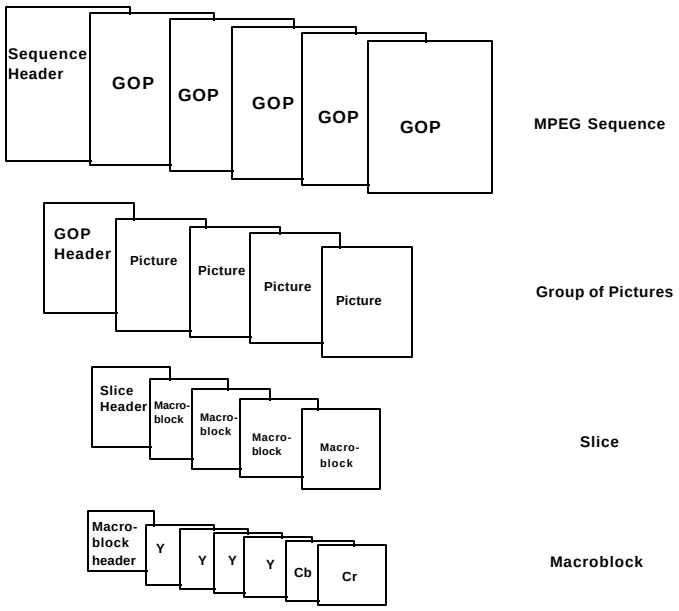

Figure 11. MPEG data stream.

P Picture Format

The **P** (Predicted) picture format introduces the concept of motion compensation. Each macroblock is coded with a vector that predicts its value from an earlier **I** or **P** frame. The decoding process copies the contents of the macroblock-sized data at the address referenced by the vector into the macroblock of the **P** frame currently being decoded. Five bits of resolution are reserved for the magnitude of the vector in each of the x and y directions, meaning that 1024 possible data blocks may be referenced by the predicted macroblock. However, eight possible magnitude ranges may be assigned to those five bits, meaning as many as 8192 macroblocks might have to be evaluated to exhaustively determine the best vector. Each evaluation might require testing as many as 384 pixels, and a further complexity is seen in performing fractional interpolation of pixels (vector motions as small as 1/2 pixel are supported). Finally, the difference between the prediction and the macroblock to be compressed may be encoded in like fashion to **I** frame encoding above.

B Picture Format

The **B** (Bidirectional prediction) picture format is calculated with two vectors. A backward vector references a macroblock-sized region in the previous **I** or **P** frame, the forward vector references a macroblock-sized region in the next **I** or **P** frame. For this reason, **I** and **P** frames are placed in the coded stream before any **B** frames that reference them.

The macroblock-sized regions referenced by the motion compensation vectors are averaged to produce the motion estimate for the macroblock being decoded. As with **P** frames, the error between the prediction and the frame being encoded is compressed and placed in the bitstream. The error factor is decompressed and added to the prediction to form the **B** frame macroblock.

Many demanding technical issues are raised by the MPEG specification. These include fast algorithms for the DCT, fast algorithms for motion vector estimation, algorithms for adaptive quantization, and decompression in environments that allow some errors.

4.4 MOTION ESTIMATION AND COMPENSATION

The coding process for **P** and **B** frames includes the motion estimator, which finds the best matching block in the available reference frames. **P** frames always use forward prediction, while **B** frames always use bidirectional prediction, also called motion-compensated interpolation, as illustrated in Figure 12 [4,12].

B frames can use forward, backward prediction, or interpolation. A block in the current frame (**B** frame) can be predicted by another block from the past reference frame (**B = A** → forward prediction), or from the future reference frame (**B** = **C** → backward prediction), or by the average of two blocks (**B** = (**A** + **C**)/2 → interpolation).

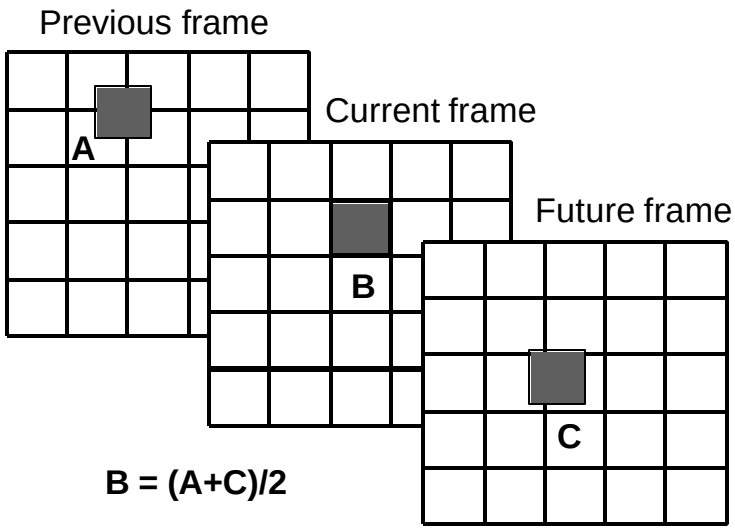

Figure 12. Motion compensated interpolated implemented in MPEG. Each block in the current frame is interpolated using the blocks from a previous and a future frame.

Motion estimation is used to extract the motion information from the video sequence. For every 16 x 16 block of **P** and **B** frames, one or two motion vectors are calculated. One motion vector is calculated for **P** and forward and backward predicted **B** frames, while two motion vectors are calculated for interpolated **B** frames.

The MPEG standard does not specify the motion estimation technique; however, block-matching techniques are likely to be used. In block-matching techniques, the goal is to estimate the motion of a block of size (n x m) in the present frame in relation to the pixels of the previous or the future frames. The block is compared with a corresponding block within a search area of size ($m + 2p$ x $n + 2p$) in the previous (or the future) frame, as illustrated in Figure 13(a). In a typical MPEG system, a match block (or a macroblock) in 16x16 pixels ($n = m = 16$), and the parameter $p = 6$ [Figure 13 (b)].

Many block-matching techniques for motion vector estimation have been developed and evaluated in the literature, such as

(i) the exhaustive search (or brute force) algorithm;
(ii) the three-step-search algorithm;
(iii) the 2-D logarithmic search algorithm;
(iv) the conjugate direction search algorithm;

(v) the parallel hierarchical 1-D search algorithm; and
(vi) the modified pixel-difference classification, layered structure algorithm.

Figure 13. The search area in block-matching techniques for motion vector estimation: (a) general case, (b) typical case for MPEG: $n = m = 16$, $p = 6$.
F - a macroblock in the current frame; G - search area in a previous (or a future) frame.

4.4.1 Cost Functions

The block matching techniques for motion estimation obtain the motion vector by minimizing a cost function. The following cost functions have been proposed in the literature:

(a) The Mean-Absolute Difference (MAD), defined as:

$$MAD\,(dx, dy) = \frac{1}{mn} \sum_{i=n/2}^{n/2} \sum_{j=-m/2}^{m/2} | F(i, j) - G(i + dx, j + dy) | \qquad [1]$$

where

F(I, j) represents a (m x n) macroblock fro m the current frame,

G(I, j) represents the same macroblock from a reference frame (past or future),

(dx, dy) a vector representing the search location.

The search space is specified by dx = { -p, +p} and dy = {-p, + p}.

For a typical MPEG system, $m = n = 16$ and $p = 6$, the MAD function becomes:

$$MAD(dx, dy) = \frac{1}{256} \sum_{i=-8}^{8} \sum_{j=-8}^{8} | F(i, j) - G(i + dx, j + dy)| \qquad [2]$$

and

$$dx = \{-6, 6\}, dy = \{-6, 6\}$$

(b) The Mean-Squared Difference (MSD) cost function is defined as:

$$MSD(dx, dy) = \frac{1}{mn} \sum_{i=-n/s}^{n/2} \sum_{i=-m/2}^{m/2} [F(i, j) - G(i + dx, j + dy)]^2 \qquad [3]$$

(c) The Cross-Correlation Function (CCF) is defined as:

$$CCF(dx, dy) = \frac{\sum_i \sum_j F(i, j) G(i + dx, j + dy)}{\left(\sum_i \sum_j F^2(i, j)\right)^{\frac{1}{2}} \left(\sum_i \sum_j G^2(i + dx, j + dy)\right)^{\frac{1}{2}}} \quad [4]$$

The mean absolute difference (MAD) cost function is considered a good candidate for video applications, because it is easy to implement in hardware. The other two cost functions, MSD, and CCF, can be more efficient, but are too complex for hardware implementations.

To reduce the computational complexity of MAD, MSD, and CCF cost functions, Gharavi and Mills have proposed a simple block matching criterion, called Pixel Difference Classification (PDC) [13]. The PDC criterion is defined as:

$$PDC(dx, dy) = \sum_i \sum_j T(dx, dy, i, j) \quad [5]$$

for (d*x*, d*y*) = {-p, p}.

T (d*x*, d*y*, I, j) is the binary representation of the pixel difference defined as:

$$T(dx, dy, i, j) = \begin{cases} 1, & |F(i, j) - G(i + dx, j + dy)| \leq t; \\ 0, & otherwise \end{cases} \quad [6]$$

where *t* is a pre-defined threshold value.

In this way, each pixel in a macroblock is classified as either a matching pixel (*T* = 1) or a mismatching pixel (*T* = 0). The block that maximizes the PDC function is selected as the best matched block.

4.4.2 Motion Vector Estimation Algorithms

The exhaustive search algorithm.

The exhaustive search algorithm is the simplest but computationally most intensive search method that evaluates the cost function at every location in the search area. If MSD cost function is used for estimating the motion vector, it would be necessary to evaluate $(2p + 1)^2$ MSE functions. For $p = 6$, it gives 169 iterations for each macroblock..

The three-step search algorithm.

The three-step search algorithm, proposed by Koga et al. [14] and implemented by Lee et al. [15] first calculated the cost function at the center and eight surrounding locations in the search area. The location that produces the smallest cost function (typically MSD function is used) becomes the center location for the next step, and the search range is reduced by half.

A three-step motion vector estimation algorithm for $p = 6$ is shown in Figure 14.

Figure 14. The three-step motion vector estimation algorithm – an example.

Step 1
In the first step, nine values for the cost function MAD (for simplification purposes denoted as M) are calculated: $M_1 = M$ (0, 0), $M_2 = M$ (3, 0), $M_3 = M$ (3, 3), $M_4 = M$ (0, 3), $M_5 = M$ (-3, 3), $M_6 = M$ (-3, 0), $M_7 = M$ (-3, -3), $M_8 = (0, -3)$, $M_9 = M$ (3, -3), as illustrated in Figure 14. Assuming that M_3 gives the smallest cost function, it becomes the center location for the next step.

Step 2
Nine new cost functions are calculated, for M_3 and eight surrounding locations, using a smaller step equal to 2. These nine points are denoted in Figure 14 as $M_{11}, M_{12}, M_{13}\ldots M_{19}$.

Step 3
In the last step, the location with the smallest cost function is selected as a new center location (in the example in Figure 14, this is M_{15}), and nine new cost functions are calculated surrounding this location: $M_{21}, M_{22}, M_{23},\ldots M_{29}$. The smallest value is the final estimate of the motion vector. In the example in Figure 14, it is M_{24}, which gives the motion vector $\{dx, dy\}$ equal to {1, 6}.

Note that the total number of computations of the cost function is: 9 x 3 – 2 = 25, which is much better than l69 in the exhaustive search algorithm.

The 2-D logarithmic search algorithm.
This algorithm, proposed by Jain and Jain [16], uses the MSD cost function and performs a logarithmic 2-D search along a virtual direction of minimum distortion (DMD) on the data within the search area. The modified version of the algorithm, described by Srinivasan and Rao [17], uses the MAD cost function, and can be described using the following steps, as illustrated in Figure 15.

Figure 15. The modified 2-D logarithmic search algorithm.

Step 1

The MAD function is calculated for dx = dy = 0, M (0, 0) and compared to the threshold (e.g., the value is 4 out of 255): M (0, 0) < T. If this is satisfied, the tested block is unchanged and the search is complete.

Step 2a

The next four cost functions are calculated, M_1(4, 0), M_2(0, 4), M_3(-4, 0), and M_4(0, -4), and their minimum is found and compared to M (0, 0):

$$M' = \min(M_1, M_2, M_3, M_4) < M(0,0).$$

If the minimum M' > M (0, 0), go to step 3; otherwise, this value is compared against the threshold, T. If M' < T, the value M' is the minimum and the search ends. Otherwise, the algorithm continues with step 2b.

Step 2b

Assuming in the previous step 2a that the minimum M' = M_1(4, 0), then the next two surrounding positions are calculated: M_5(4, 4), and M_6(4, -4), as indicated in Figure 15. The tests for minimum and threshold are performed again and, if the minimum is found, the procedure is complete. Otherwise, step 3 continues.

Step 3

Assuming that the new minimum location is M_5(4, 4), a similar search procedure (steps 2a and 2b) is continued, except the step is divided by 2. In Figure 15, the new minimum becomes M(2, 4).

Step 4

The step is further reduced by 2, and the final search (steps 2a and 2b) is performed. The minimum (dx, dy) is found. In Figure 15 it is (1, 5).

For p = 6, this algorithm requires maximum 19 cost function calculations, as shown in Figure 15.

Figure 16. A principle of the conjugate direction search algorithm (one-at-a-time search).

The conjugate direction search algorithm.
This algorithm for motion vector estimation, proposed by Srinivasan and Rao [17], is an adaptation of the traditional iterative conjugate direction search method. This method can be implemented as one-at-a-time search method, as illustrated in Figure 16.

In Figure 16, direction of search is parallel to one of coordinate axes, and each variable is adjusted while the other is fixed. This method has been adapted for motion vector estimation [17], as illustrated in Figure 17. The algorithm consists of the following three steps:

Figure 17. The conjugate direction search method for motion vector estimation.

Step 1
Values of the cost function MAD in the dx direction are calculated until the minimum is found. The calculation is as follows: (a) $M(0, 0)$, $M(1, 0)$, and $M(-1, 0)$. (b) If $M(1, 0)$ is the minimum, $M(2, 0)$ is computed and evaluated, and so on. This step is complete when a minimum in the dx direction is found [in Figure 17, the minimum is $M(2, 0)$].

Step 2
The search now continues in the dy direction by calculating cost functions $M(2, -1)$ and $M(2, 1)$. A minimum in the dy direction is then found at $M(2, 2)$ in Figure 17.

Step 3
The direction of search is now the vector connecting the starting point (0, 0) and the obtained minimum (2, 2). The following cost functions are calculated and evaluated next: $M(1, 1)$ and $M(3, 3)$, and so on, until a minimum in this direction is found. In the example in Figure 17, the minimum is $M(4, 4)$, and the obtained motion vector is dx = 4 and dy = 4.

It may happen that the dx and dy vectors, obtained in steps 2 and 3, do not constitute a square as given in Figure 17. In that case, the nearest grid points on the direction joining (0, 0) and the obtained minimum point are selected.

The parallel hierarchical 1-D search algorithm (PHODS).
The PHODS algorithm, proposed by Chan *et al*. [18], reduces the number of blocks to be searched. In this algorithm, the 2-D motion vector for each block is represented as two 1-D motion vectors, one in horizontal and one in vertical direction. Both vectors are searched in parallel. The algorithm uses MAD or MSD cost function and can be described using the following steps as illustrated in Figure 18.

Figure 18. The parallel hierarchical 1-D search algorithm – an example.

Step 1
The cost function is further calculated in the search center (M_0), and in two additional positions in both directions (M_1 and M_2 in horizontal direction, and M_3 and M_4 in vertical direction). The positions with the smallest cost functions in both directions become the search centers in the next step, and the distance between the candidate blocks is reduced. M_1 gives the minimum in horizontal direction, while M_4 in the vertical direction. They are new search centers for the next step.

Step 2
The cost function is further calculated in M_1, M_{21}, and M_{22} horizontal locations and M_4, M_{23} and M_{24} vertical locations. New minimum locations are found as M_{22} in horizontal and M_{24} in vertical direction.

Step 3
In the last step, the search step is further reduced to 1, and the cost function is calculated in horizontal locations M_{22}, M_{31}, and M_{32}, and vertical locations M_{24}, M_{33}, and M_{34}. Assuming the M_{31} is the minimum in horizontal direction and M_{33} in vertical direction, the estimation motion vector becomes dx = -2 and dy = -4.

The modified pixel-difference classification, layered structure algorithm (MPDC-LSA).
This algorithm, proposed by Chan et al. [18], is a block-matching algorithm based on a modified PDC function, which gives low computational complexity. The algorithm is based on the MPDC criterion, where a block is classified as a matching block if all its pixels are matching pixels with respect to the defined threshold. The MPDC criterion is defined as

$$MPDC \quad (dx, dy) = \begin{cases} 1, & if \quad PDC \quad (dx, dy) = n^2; \\ 0, & otherwise \end{cases} \qquad [7]$$

The search procedure and experimental results are presented by Chan et al. [18].

Using a block-matching motion estimation technique, the best motion vector(s) is found which specifies the space distance between the actual and the reference macroblocks. The macroblock in the current frame is then predicted based on a macroblock in a previous frame (forward prediction), a macroblock in a future frame (backward prediction), or using interpolation between macroblocks in a previous and a future frame. A macroblock in the current frame $F(i, j)$ is predicted using the following expression:

$$F_p(i, j) = G(i\ dx,\ j + dy) \qquad [8]$$

for $(i, j) = \{-8, 8\}$.

$F_p(i, j)$ is the predicted current macroblock, $G(i, j)$ is the same macroblock in a previous/future frame, and (dx, dy) is the estimated motion vector.

For interpolated frames, a macroblock in the current frame $F(i, j)$ is predicted using the following formula

$$F_p(i, j) = \tfrac{1}{2}[G_1(i + dx_1) + G_2(i + dx_2, j + dy_2)] \qquad [9]$$

`

for $(i, j) = \{-8, 8\}$,

$G_1(i, j)$ is the same macroblock in a previous frame, (dx_1, dy_1) is the corresponding motion vector, $G_2(i, j)$ is the same macroblock in a future frame, and (dx_2, dy_2) is its corresponding motion vector.

The difference between predicted and actual macroblocks, called the error terms $E(i, j)$, is then calculated using the following expression:

$$E(i, j) = F(i, j) - F_p(i, j) \quad [10]$$

for $(i, j) = \{-8, 8\}$.

4.5 MPEG AUDIO ENCODER AND DECODER

The MPEG standard also covers audio compression. MPEG uses the same sampling frequencies as compact disc digital audio (CD-DA) and digital audio tapes (DAT). Besides these two frequencies, 44.1 KHz and 48 KHz, 32 KHz are also supported, all at 16 bits. The audio data on a compact disc, with two channels of audio samples at 44.1 KHz with 16 bits/sample, require a data rate of about 1.4 Mbits/s [19]. Therefore, there is a need to compress audio data as well.

Existing audio compression techniques include μ-law and Adaptive Differential Pulse Code Modulation (ADPCM), which are both of low complexity, low compression ratios, and offer medium audio quality. The MPEG audio compression algorithm is of high complexity, but offers high compression ratios and high audio quality. It can achieve compression ratios ranging from 5:1 to 10:1.

The MPEG audio compression algorithm is comprised of the following three operations:

(i) The audio signal is first transformed into the frequency domain, and the obtained spectrum is divided into 32 non-interleaved subbands.

(ii) For each subband, the amplitude of the audio signal is calculated, and the noise level determined by using a "psychoacoustic model." The psychoacoustic model is the key component of the MPEG audio encoder and its function is to analyze the input audio signal and determine where in the spectrum the quantization noise should be masked.

(iii) Finally, each subband is quantized according to the audibility of quantization noise within that band.

The MPEG audio encoder and decoder are shown in Figure 19 [7,10,11,19].

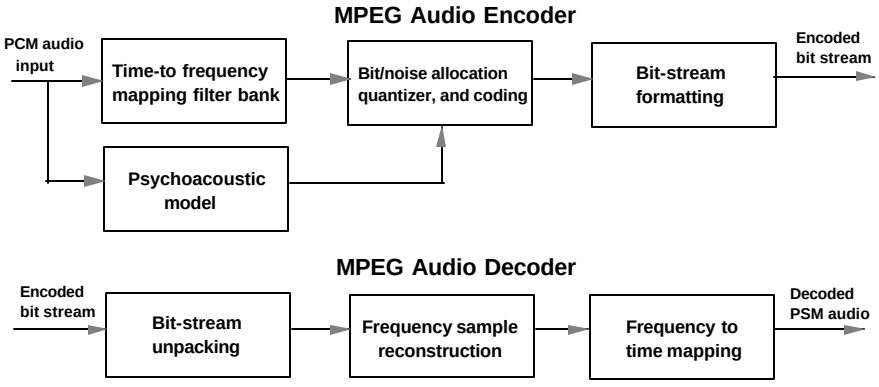

Figure 19. Block diagrams of MPEG audio encoder and decoder.

The input audio stream simultaneously passes through a filter bank and a psychoacoustic model. The filter bank divides the input into multiple subbands, while the psychoacoustic model determines the signal-to-mask ratio of each subband. The bit or noise allocation block uses the signal-to-mask ratios to determine the number of bits for the quantization of the

subband signals with the goal to minimize the audibility of the quantization noise. The last block performs entropy (Huffman) encoding and formatting the data. The decoder performs entropy (Huffman) decoding, then reconstructs the quantized subband values, and transforms subband values into a time-domain audio signal.

The MPEG audio standard specifies three layers for compression: layer 1 represents the most basic algorithm and provides the maximum rate of 448 Kbits/s, layers 2 and 3 are enhancements to layer 1 and offer 384 Kbits/s and 320 Kbits/s, respectively. Each successive layer improves the compression performance, but at the cost of the greater encoder and decoder complexity.

A detailed description of audio compression principles and techniques is reported elsewhere [10,11,19].

4.6 MPEG INTERLEAVED A/V DATA STREAM

The MPEG standard specifies a syntax for the interleaved audio and video data streams. An audio data stream consists of frames, which are divided into audio access units. Audio access unit consists of slots, which can be either four bits at the lowest complexity layer (layer 1), or one byte at layers 2 and 3. A frame always consists of a fixed number of samples. Audio access unit specifies the smallest audio sequence of compressed data that can be independently decoded. The playing times of the audio access units of one frame are 8 ms at 48 KHz, 8.7 ms at 44.1 KHz, and 12 ms at 32 KHz [7]. A video data stream consists of six layers, as shown in Table 8.

Table 8. Layers of MPEG Video Stream Syntax

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

At the beginning of the *sequence layer* there are two entries: the constant bit rate of a sequence and the storage capacity that is needed for decoding. These parameters define the data buffering requirements. A sequence is divided into a series of GOPs. Each GOP layer has at least one **I** frame at the first frame in GOP, so random access and fast search are enabled. GOPs can be of arbitrary structure (**I**, **P,** and **B** frames) and length. The GOP layer is the basic unit for editing an MPEG video stream.

The *picture layer* contains a whole picture (or a frame). This information consists of the type of the frame (**I**, **P,** or **B**) and the position of the frame in the display order.

The bits corresponding to the DCT coefficients and the motion vectors are contained in the next three layers: *slice*, *macroblock*, and *block* layers. The block is a (8x8) DCT unit, the macroblock is a (16x16) motion compensation unit, and the slice is a sting of macroblocks of arbitrary length. The slice layer is intended to be used for re-synchronization during a frame decoding when bit errors occur.

5. THE XYZ VIDEO COMPRESSION ALGORITHM

The XYZ motion video compression algorithm relies on a different principle for compression of temporal information than the MPEG and H.261/H.263 standards. While the MPEG and

H.261/H.263 strategies look for motion vectors to represent a frame being compressed, the XYZ strategy more closely resembles the technique adopted by both MPEG and JPEG for intraframe compression.

A continuous tone image can be represented as a two-dimensional array of pixel values in the spatial domain. The Forward Discrete Cosine Transform (FDCT) converts the two-dimensional image from spatial to frequency domain. In spatial representation the energy distribution of pixels is uniform, while in the frequency domain the energy is concentrated into a few low-frequency coefficients.

Pixels in full-motion video are also correlated in the temporal domain, and the FDCT will concentrate the energy of pixels in the temporal domain just as it does in the spatial domain. The XYZ video compression is based on this property.

5.1 THE XYZ ENCODER

The XYZ video compression algorithm is based on the three-dimensional DCT (3D DCT). This algorithm takes a full-motion digital video stream and divides it into groups of 8 frames. Each group of 8 frames is considered as a three-dimensional image, where X and Y are spatial components and Z is the temporal component. Each frame in the image is divided into 8x8 blocks (like JPEG), forming 8x8x8 cubes, as illustrated in Figure 20. Each 8x8x8 cube is then independently encoded using the three blocks of the XYZ video encoder: 3D DCT, Quantizer, and Entropy encoder [WF95]. The block diagram of the XYZ encoder is shown in Figure 21.

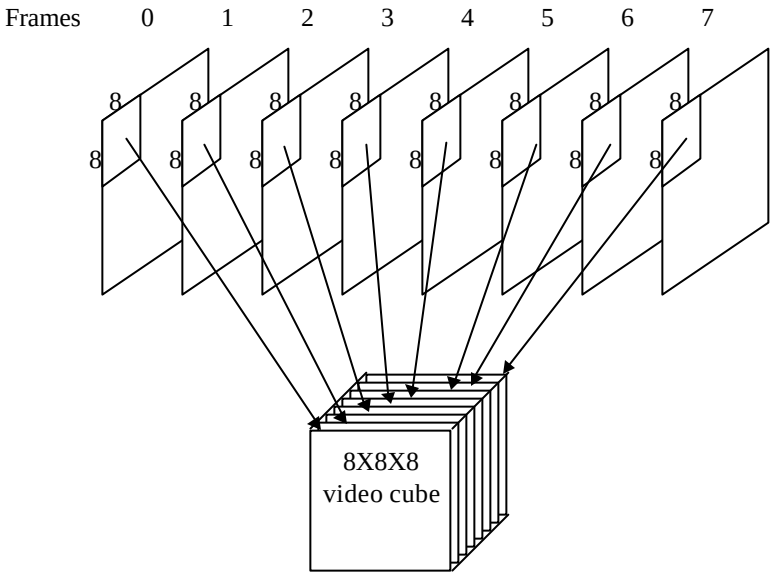

Figure 20. Forming 8x8x8 video cube for XYZ compression.

The original unsigned pixel sample values, typically in the range [0,255], are first shifted to signed integers, say in the range [-128,127]. Then each 8x8x8 cube of 512 pixels is transformed into the frequency domain using the Forward 3D DCT:

$$F(u,v,w) = C(u)C(v)C(w) * \sum_{x=0}^{7}\sum_{y=0}^{7}\sum_{z=0}^{7} f(x,y,z) * \frac{\cos((2x+1)up)}{16}\frac{\cos((2y+1)vp)}{16}\frac{\cos((2z+1)wp)}{16}$$

where:
x,y,z are index pixels in pixel space,
f(x,y,z) is the value of a pixel in pixel space,
u,v,w are index pixels in DCT space,
F(u,v,w) is a transformed pixel value in DCT space, and

$$C(i) = \frac{1}{\sqrt{2}} \quad \text{for i = 0} \qquad C(i) = 1 \quad \text{for i > 0}$$

The transformed 512-point discrete signal is a function in three dimensions, and contains both spatial and temporal information. Most of the energy is contained in few low-frequency coefficients, while the majority of the high-frequency coefficients have zero or near-zero values.

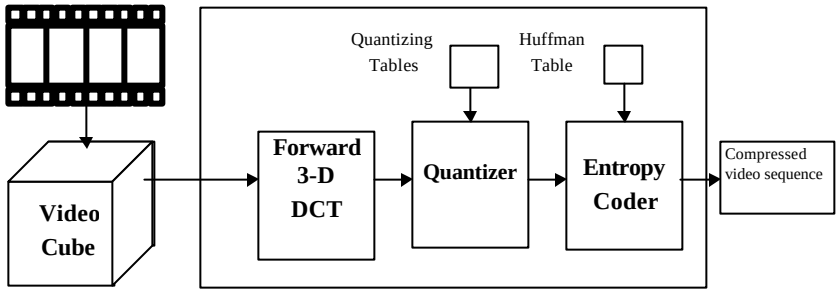

Figure 21. Block diagram of the XYZ encoder.

In the next step, all 512 DCT coefficients are quantized using a 512-element quantization table. Quantization introduces minimum error while increasing the number of zero-value coefficients. Quantization may also be used to discard visual information to which the human eye is not sensitive. Quantizer tables may be predefined, or adaptive quantizers may be developed and transmitted with the compressed data.

Quantization is performed according to the following equation:

$$F_q(u,v,w) = \left\lfloor \frac{F(u,v,w)}{Q(u,v,w)} \right\rfloor$$

where:
F(u,v,w) are the elements before the quantization,
$F_{q(}$u,v,w) are the quantized elements, and
Q(u,v,w) are the elements from the quantization table.

Each quantizer Q(u,v,w) is in the range [1,1024]. The result of the quantization operation is a collection of smaller-valued coefficients, a large number of which are 0. These coefficients are then converted into a compact binary sequence using an entropy coder (in this case, a Huffman coder).

The entropy coding operation starts with reordering the coefficients in descending order of expected value. This sequence has the benefit of sequentially collecting the largest number of zero-valued coefficients. The run-lengths of zero coefficients is computed, and the alphabet of symbols to be encoded becomes the run-length of zeros appended to the length of the non-zero coefficient. This binary sequence represents the compressed 8x8x8 block.

Figure 22 illustrates an example of encoding a video cube (eight frames of 8x8 pixels) using the XYZ compression algorithm. Figure 22 shows the original video cube, Figure 23a shows the DCT coefficients after the 3D DCT, and Figure 23b presents the quantized coefficients. Note that the largest quantized coefficient is Fq(0,0,0), which carries the crucial information on the video cube, while the majority of quantized coefficients are zero.

Figure 22. An example of encoding an 8x8x8 video cube – original pixels in video cube.

Figure 23. An example of encoding an 8x8x8 video cube.
(a) DCT coefficients, after 3-D DCT, and (b) quantized DCT coefficients.

5.2 THE XYZ DECODER

In XYZ decoding, the steps from the encoding process are inverted and implemented in reverse order, as shown in Figure 24.

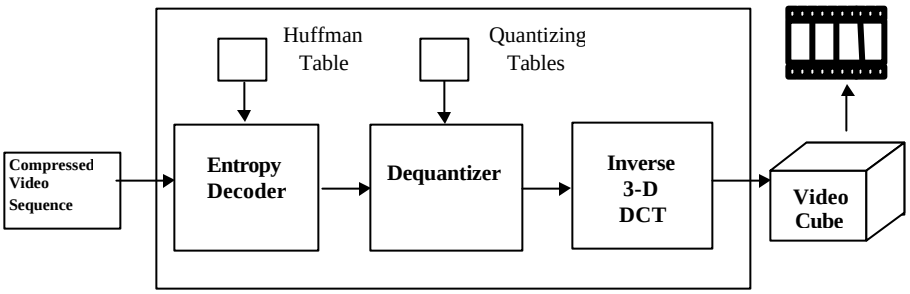

Figure 24. Block diagram of the XYZ decoder.

First the compressed data stream is Huffman-decoded. This data stream is now composed of the coding alphabet symbols of run-length and VLC lengths alternated with the VLC representation of the non-zero coefficient. The decoded data is run-length expanded and converted into a stream of quantized coefficients. These quantized coefficients are resequenced into XYZ video cubes of quantized coefficients.

The quantized coefficients are dequantized according to the following equation:

$$F'(u,v,w) = F_q(u,v,w) * Q(u,v,w)$$

where F'(u,v,w) is a dequantized coefficient.

The three-dimensional inverse DCT (3-D IDCT) is implemented on the dequantized coefficients in order to convert video from the frequency domain into the spatial/temporal domain. The 3-D IDCT equation is defined as

$$f'(x,y,z) = \sum_{u=0}^{7}\sum_{v=0}^{7}\sum_{w=0}^{7} C(u)C(v)C(w) * F'(u,v,w) * \frac{\cos((2x+1)up)}{16}\frac{\cos((2y+1)vp)}{16}\frac{\cos((2z+1)wp)}{16}$$

where f'(x,y,z) is the value of a pixel in pixel space.

After the pixels have been transformed in spatial/temporal representation, they are shifted back to the original range [0,255]. Finally, the video cubes are reorganized into frames of video data ready for playback.

Figure 25 illustrates an example of the XYZ decompression, applied on the same 8x8x8 video cube in Figures 22 and 23.

Figure 25. An example of decoding the 8x8x8 video cube from Figure 22.
(a) DCT coefficients after dequantization, (b) decompressed pixels after inverse 3-D DCT.

5.3 COMPARISON WITH MPEG STANDARD

We compared the XYZ compression algorithm with the MPEG standard. Motion estimation in MPEG is performed by 2D logarithmic search and by exhaustive search of a region 16 pixels wide. First frame (frame 0) is compressed as **I** frame using MPEG-recommended quantizer. Frame 7 is compressed as P frame using forward motion prediction. Error terms are calculated and encoded using the DCT. Frames 1 to 6 are compressed as bidirectional B-frames. Motion estimation is done using bidirectional prediction. Four experiments were performed using MPEG: two including error correction with two different search algorithms, and two with no error correction for both search algorithms.

We applied XYZ compression for five sets of quantization tables, referred to as QT1 to QT5 in [20,21,22]. The quantization tables are selected so that QT1 tables contain the smallest coefficients, thus achieving the best quality of the video and the lowest compression ratio. On the other hand, the QT5 tables have the largest coefficients, thus producing the highest compression ratio and the lowest video quality. The results are summarized in Table 9.

Table 9. Comparison of XYZ and MPEG Video Compression Algorithms

Video Compression Technique	**Compression Ratio**	**Normalized RMS Error**	**Execution Time [min]** (8 frames, 320x240)
XYZ (QT1)	34.5	0.079	6.45
XYZ (QT2)	57.7	0.097	6.45
XYZ (QT3)	70.8	0.105	6.45
XYZ (QT4)	101.7	0.120	6.45
XYZ (QT5)	128.1	0.130	6.45
MPEG Logarithmic Search and Error Correction	11.0	0.080	21.35
MPEG Exhaustive Search and Error Correction	15.6	0.080	163.0
MPEG Logarithmic Search and No Error Correction	27.0	0.140	21.35
MPEG Exhaustive Search and No Error Correction	32.9	0.125	163.0

6. CONCLUSIONS

This chapter presented two commonly used video compression standards, px64 Kbps (or H261/263 standard) and MPEG. We also presented the XYZ video compression algorithm, based on 3D-DCT transformation.

The following conclusions can be made [20,21]:

- The XYZ video compression algorithm gives significantly better compression ratios than the MPEG algorithm for the same quality of video. For example, XYZ result 1 and

MPEG result 1 (see Table 9) give similar NRMS errors (0.079 and 0.08, respectively) and reconstructed sequences show similar image quality. However, the XYZ algorithm provides much higher compression ratio (34.5 vs 15.6).

- For similar compression ratios, the XYZ video compression algorithm gives much better quality of the decompressed video than the MPEG algorithm. For example, XYZ result 1 and MPEG result 4 (see Table 9) give similar compression ratios (34.5 and 32.9, respectively), but XYZ algorithm gives much better quality (NRMS error for XYZ is 0.079, while for MPEG it is 0.125).

- The obtained results suggest that XYZ video compression algorithm is faster than the MPEG algorithm (including both compression and decompression).

- The XYZ results 4 and 5 (Table 9) suggest that very high compression ratios (greater than 100) can be achieved using the XYZ video compression algorithm, while the NRMS error is still kept relatively small (in the range from 0.120 to 0.130). In this case, for videos with the fast camera movement the visual artifacts are significant. However, the algorithm gives very good results for videos with little movement, which is the case in videoconferencing applications.

- Finally, the MPEG technique is based on three different algorithms: one for **I**, another for **P**, and the third algorithm for **B** frames. MPEG is also an asymmetrical algorithm, requiring a complex encoder and a simple decoder. On the other hand, the XYZ technique applies only one algorithm to all frames and is a symmetrical algorithm requiring the same complexity for both encoder and decoder. This fact is beneficial for VLSI implementation of the algorithm, which is described in [21].

REFERENCES

1. B. Furht, "A Survey of Multimedia Techniques and Standards, Part I: JPEG Standard," *Real-Time Imaging Journal*, Vol. 1, No. 1, pp. 49-67, 1995.
2. B. Furht, "A Survey of Multimedia Compression Techniques and Standards," Part II: Video Compression," *Real-Time Imaging Journal*, Vol. 1, pp. 319-337, 1995.
3. M. Liou, "Overview of the Px64 Kbits/s Video Coding Standard," *Communications of the ACM*, Vol. 34, No. 4, pp. 59-63, 1991.
4. R. Aravind, G.L. Cash, D.C. Duttweller, H-M. Hang, and A. Puri, "Image and Video Coding Standards," *AT&T Technical Journal*, Vol. 72, pp. 67-88, 1993.
5. M. Ghanbari, "An Adapted H.261 Two-Layer Video Codec for ATM Networks," *IEEE Transactions on Communications*, Vol. 40, No. 9, pp.1481-1490, 1992.
6. H.-C. Huang, H.-H. Huang, and J.-L. Wu, "Real-Time Software-Based Video Coder for Multimedia Communication Systems," *Journal of Multimedia Systems*, Vol. 1, pp. 110-119, 1993.
7. R. Steinmetz, "Data Compression in Multimedia Computing - Standards and Systems," Part I and II, *Journal of Multimedia Systems*, Vol. 1, pp. 166-172 and 187-204, 1994.
8. G. LeGall, "MPEG: A Video Compression Standard for Multimedia Applications," *Communications of the ACM, Vol*. 34, No. 4, pp. 45-68, 1991.
9. W.B. Pennenbaker and J.L. Mitchell, "JPEG Still Image Data Compression Standard," Van Nostrand Reinhold, New York, 1993.
10. V. Bhaskaran and K. Konstantinides, "Image and Video Compression Standards – Algorithms and Architectures," Kluwer Academic Publishers, Norwell, MA, 1995.
11. J. L. Mitchell, W.B. Pennenbaker, C.E. Fogg, and D.J. LeGall, "MPEG Video Compression Standard," Chapman & Hall, New York, 1996.

12. B. Furht, J. Greenberg, and R. Westwater, "Motion Estimation Algorithms for Video Compression," Kluwer Academic Publishers, Norwel, MA, 1997.
13. H. Gharavi and M. Mills, "Block Matching Motion Estimation Algorithms – New Results," *IEEE Transactions on Circuits Systems*, Vol. 37, pp. 649-651, 1990.
14. J. Koga, K. Iinuma, A. Hirani, Y. Iijima, and T. Ishiguro, "Motion Compensated Interframe Coding for Video Conferencing," Proceedings of the National Telecommunications Conference, pp. G5.3.1-5.3.5, 1981.
15. W. Lee, Y. Kim, R.J. Gove, and C.J. Read, "Media Station 5000: Integrating Video and Audio," *IEEE MultiMedia*, Vol. 1, No. 2, pp. 50-61, 1994.
16. J.R. Jain, and A.K. Jain, "Displacement Measurement and Its Application in Interframe Image Coding," *IEEE Transactions on Communications*, Vol. 29, pp. 1799-1808, 1981.
17. R. Srinivasan and K.R. Rao, "Predictive Coding Based on Efficient Motion Estimation," *IEEE Transactions on Communications*, Vol. 33, pp. 888-896, 1985.
18. E. Chan, A.A. Rodriguez, R. Gandhi, and S. Panchanathan, "Experiments on Block-Matching Techniques for Video Coding," *Journal of Multimedia Systems*, Vol. 2, No. 5, pp. 228-241, 1994.
19. D. Y. Pen, "Digital Audio Compression," *Digital Technical Journal*, Vol. 5, No. 2, pp. 28-40, 1993.
20. R. Westwater and B. Furht, "The XYZ Algorithm for Video Compression of Full-Motion Video," *Real-Time Imaging Journal*, Vol. 2, pp. 19-34, 1996.
21. R. Westwater and B. Furht, "Real-Time Video Compression – Techniques and Algorithms," Kluwer Academic Publishers, Norwell, MA, 1997.
22. R. Westwater and B. Furht, "Three-Dimensional DCT Video Compression Technique Based on Adaptive Quantizers," Second IEEE International Conference on Engineering of Complex Computer Systems, Montreal, Canada, October 1996.