Chapter 3

Multimedia Systems Technology: Coding and Compression

3.1 Introduction

In multimedia system design, *storage* and *transport* of information play a significant role. Multimedia information is inherently voluminous and therefore requires very high storage capacity and very high bandwidth transmission capacity. For instance, the storage for a video frame with 640×480 pixel resolution is 7.3728 million bits, if we assume that 24 bits are used to encode the luminance and chrominance components of each pixel. Assuming a frame rate of 30 frames per second, the entire 7.3728 million bits should be transferred in 33.3 milliseconds, which is equivalent to a bandwidth of 221.184 million bits per second. That is, the transport of such large number of bits of information and in such a short time requires high bandwidth.

There are two approaches that are possible - one to develop technologies to provide higher bandwidth (of the order of Gigabits per second or more) and the other to find ways and means by which the number of bits to be transferred can be reduced*, without compromising the information content.* It amounts to saying that we need a transformation of a string of characters in some representation (such as ASCII) into a new string (e.g., of bits) that contains the same information but whose length must be as small as possible; i.e., *data compression.* Data compression is often referred to as coding, whereas coding is a general term encompassing any special representation of data that achieves a given goal. For example, reducing the number of bits is by itself a goal.

In 1948, Claude Shannon, the father of Information Theory, introduced

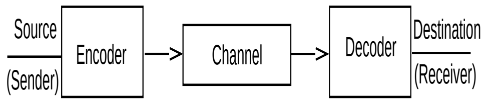


Fig. 3.1. Shannon's Communication Systems Model

a concept called *entropy* to measure the *information content* of a source[1]. Shannon proposed a communication system model (shown pictorially in Fig. 3.1) which addresses two basic issues:

- How can a communication system *efficiently transmit the information* that a source produces?
- How can a communication system achieve *reliable communication* over a noisy channel?

The first issue relates to *compression* and the second relates to *error control*. In both cases, there is a *transformation of number of bits* that represent the information produced by a source. These two aspects are also known as *source coding* and *channel coding*. Information theory is the study of efficient coding and its consequences in the form of speed of transmission and probability of error.

In this chapter we focus on source coding (compression) and standards for compression. Information coded at the source end has to be correspondingly decoded at the receiving end. Coding can be done in such a way that the *information content* is not lost; that means it can be recovered fully on decoding at the receiver. However, media such as image and video (meant primarily for human consumption) provide opportunities to encode more efficiently but with a loss.

Coding (consequently, the compression) of multimedia information is subject to certain quality[1] constraints. For example, the quality of a picture should be the same when coded and, later on, decoded. Dealing with perceptual quality it is possible for one to design a coding method that may be lossy. By lossy coding we mean that there is information loss, but the loss does not affect the perceptual quality. In practice, one or more parameters related to the source may be used in designing coding schemes that result in lossy compression. Also, the complexity and the execution time of the techniques used for coding and decoding should be minimal. These constraints

[1]By quality, we mean perceptual quality.

are related to the nature of the application(live or orchestrated) as well. It should be noted that all these requirements are based on the characteristics of human perception of different media. For example, for *interactive* multimedia application such as videoconferencing, the end-to-end delay should be less than 150 milliseconds. In the context of current high-speed transmission technologies, such an end-to-end delay requirement would imply that compression and decompression, if any, should be contained within 50 milliseconds. Similarly, multimedia applications that are *retrieval* based, such as on-demand video service, require that random access to single images and audio frames be possible under 500 milliseconds.

Coding and compression techniques are critical to the viability of multimedia at both the storage level and at the communication level. Some of the multimedia information has to be coded in continuous (time dependent) format and some in discrete (time independent) format. In multimedia context, the *primary motive* in coding is *compression*. By nature, the audio, image, and video sources have built-in redundancy, which make it possible to achieve compression through appropriate coding. As the image data and video data are voluminous and act as prime motivating factors for compression, our references in this chapter will often be to images even though other data types can be coded (compressed).

3.2 What is Image Compression?

Image data can be compressed without significant degradation of the visual (perceptual) quality because images contain a high degree of:

- spatial redundancy, due to correlation between neighboring pixels (this is also referred to as statistical redundancy),
- spectral redundancy, due to correlation among color components, and
- psycho-visual redundancy, due to perceptual properties of the human visual system.

The higher the redundancy, the higher the achievable compression. This is what we referred to as *source coding* earlier in the communication system of Shannon. A typical image compression system (or source encoder) consists of a transformer, quantizer, and a coder, as illustrated in Fig. 3.2.

Transformer applies a one-to-one transformation to the input image data. The output of the transformer is an image representation which is

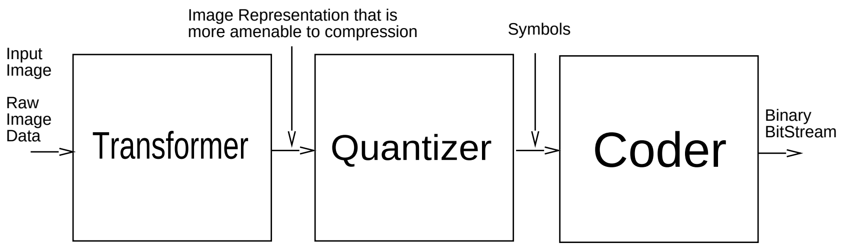


Fig. 3.2. Typical Image Compression System

more amenable to efficient compression than the raw image data. Unitary mappings such as Discrete Cosine Transform, which pack the energy of the signal to a small number of coefficients, is a popular method with image compression standards.

Quantizer generates a limited number of symbols that can be used in the representation of the compressed image. Quantization is a many-to-one mapping which is irreversible. It can be performed by scalar or vector quantizers. *Scalar quantization* refers to element-by-element quantization of data and *vector quantization* refers to quantization of a block at a time.

Coder assigns a code word, a binary bit-stream, to each symbol at the output of the quantizer. The coder may employ *fixed-length* or *variable-length* codes. Variable Length Coding (VLC), also known as entropy coding, assigns code words in such a way as to minimize the average length of the binary representation of the symbols. This is achieved by assigning shorter code words to more probable symbols, which is the fundamental principle of entropy coding.

Different image compression systems are based on different combinations of transformer, quantizer, and coder. Image compression systems can be broadly classified as:

- **Lossless:** Compression systems, which aim at minimizing the bit-rate of the compressed output without any distortion of the image. The decompressed bit-stream is identical to the original bit-stream. This method is used in cases where accuracy of the information is essential. Examples of such situations are computer programs, data, medical imaging, and so on. Lossless compression systems are also referred to as bit-preserving or reversible compression systems.

- **Lossy:** Compression systems, which aim at obtaining the best possible *fidelity* for a given bit-rate (or minimizing the bit-rate to achieve a given fidelity measure). Such systems are suited for video and audio.

The transformation and coding stages are *lossless*. However, the quantization is *lossy*.

3.3 Taxonomy of Compression Techniques

Based on the *lossless* or *lossy* compression, the encoding is broadly classified as *entropy encoding* (leading to lossless compression) and *source encoding* (leading to lossy compression).

It is interesting to note that the term *entropy encoding* refers to all those coding and compression techniques which do not take into account the nature of the information to be compressed. In other words, entropy encoding techniques simply treat all data as a sequence of bits and ignores the semantics of the information to be compressed.

On the contrary, the *source encoding* takes cognizance of the type of the original signal. For example, if the original signal is audio or video, source encoding uses their inherent characteristics in achieving better compression ratio. Normally, it is expected that source encoding will produce better compression compared to entropy encoding techniques. But, of course, the actual degree of success will depend on the semantics of the data itself and may vary from application to application. The designer of a compression system based on source encoding has the choice to render it lossless or lossy.

All of this aside, in practical systems and standards, both entropy encoding and source encoding are usually combined for better effect in compression.

Let us understand the principle behind some of the coding techniques. For an exhaustive treatment of all the techniques, the reader may want to refer to [4, 5, 6, 7].

3.4 Entropy Encoding Techniques

3.4.1 Run-length Encoding

In this technique, the sequence of image elements (or pixels in a scan line) x_1, x_2, ... ,x_n is mapped into a sequence of pairs (c_1,l_1), (c_2,l_2), ..., (c_k,l_k), where c_i represents a color (or intensity) and l_i the length of the i^{th} run (sequence of pixels with equal intensity).

Example: Let digits 1, 2, 3 represent Red, Green, and Blue. These will correspond to c_i. Let a scan line be of length 35 consisting of

11111111111333333333322222222211111

as x_i. Then, the run-length encoded stream will be the series of tuples (1,11), (3,10), (2,9), and (1,5), where 11,10,9,5 are the l_i.

3.4.2 Repetition Suppression

In this technique, a series of n successive occurrences of a specific character is replaced by a special character called flag, followed by a number representing the repetition count. Normal applications are suppression of 0s in a data file or a bitmap file and suppression of blanks in a text or program file.

Example: Consider a sequence of digits in a data file which looks like the following:

98400000000000000000000000000000000.

When we use the repetition sequence suppression method, the sequence will look like 984f32. The savings is obvious and is dependent upon the contents.

3.4.3 Pattern Substitution

Pattern substitution is a form of a statistical coding method. The idea here is to substitute an oft-repeated pattern by a code. If we are trying to compress the book Hound of Baskervilles, the name *Sherlock Homes* can be substituted by S*, the name *Watson* by W*, and the expression *Elementary my dear Watson* by E*. Obviously, frequent patterns will use shorter codes for better compression.

Example: Consider the lines:

This book is an exemplary example of a book on multimedia and networking. Nowhere else will you find this kind of coverage and completeness. This is truly a one-stop-shop for all that you want to know about multimedia and networking.

If we simply count, there are a total of 193 characters without counting blanks and 232 with blanks. If we group words such as a, about, all, an, and, for, is, of, on, that, this, to, and will, they occur 2, 1, 1, 1, 3, 1, 2, 2, 1, 1, 3,1, and 1 times, respectively. All of them have a blank character on either side, unless when they happen to be the first word or last word of a sentence. The sentence delimiter *period* is always followed by a blank character. The words *multimedia* and *networking* appear twice each. With all this knowledge about the text, we can develop a substitution table that will be very effective. Notice that there is no loss and the coding is reversible. Let us represent the group of words that we identified for the text under consideration by 1, 2, 3, 4, 5, 6, 7, 8, 9, +, &, =, and #. Let us also substitute multimedia by m* and networking by n*. The resulting coded string will be:

&	b	o	o	k	7	4	e	x	e	m	p	l	a	r	y
sp	e	x	a	m	p	l	e	8	1	b	o	o	k	9	m
*	5	n	*	.	N	o	w	h	e	r	e	sp	e	l	s
e	#	y	o	u	sp	f	i	n	d	&	k	i	n	d	8
c	o	v	e	r	a	g	e	5	c	o	m	p	l	e	t
e	n	e	s	s	.	&	7	t	r	u	l	y	1	o	n
e	-	s	t	o	p	-	s	h	o	p	6	3	+	y	o
u	sp	w	a	n	t	=	k	n	o	w	2	m	*	5	n
*	.														

That is a total of 129 characters and 33.16% compression.

3.4.4 Huffman Coding

Huffman coding is based on the frequency of occurrence of a character (or an octet in the case of images). The principle is to use a lower number of bits to encode the character that occurs more frequently. The codes are stored in a codebook. The codebook may be constructed for every image or for a set of images, when applied to still or moving images. In all cases, the codebook should be transferred to the receiving end so that decoding can take place.

Example: In the sentences given in the example for *pattern substitution*, the occurrence of the alphabets are as follows:

```
Character: a b c d e Frequency: 15 3 2 7 18 Code: 1100 0101011
0010011 01000 1011

Character:        f         g         h         i         j
Frequency:        4         3         6        14         0
Code:       0001100   0101000    101010     00111  00000111

Character:        k         l         m         n         o
Frequency:        6        11         7        16        21
Code:        101011      1000     00010      1111      1110

Character:        p         q         r         s         t
Frequency:        5         0         7        10        15
Code:       0100111 000011000     10100      0011    001000
Character:        u         v         w         x         y
Frequency:        6         1         6         2         4
Code:       0000111   0000111    110101   0100100   0000011

Character:        z         .        sp
Frequency:        0         3        39
Code:       0000100   0000100      0111
```

If they are coded as per the code given in the code-book (Table above), the binary bit stream will be:

```
001000 101010 00111 0011 0111 00111 0011 0111 1100 1111
..T... ..h... ..i.. ..s. .sp. ..i.. ..s. .sp. ..a. ..n.
0111
.sp.   ....  and so on.....
```

The total size of the bit stream will be 1124 bits as compared to 1856 bits (232 x 8) of the original text; a compression of 39.44%.

3.5 Source Encoding Techniques

Source encoding is based on the content of the original signal and hence it is rightly termed as semantic-based coding. The compression that can be achieved using source coding may be very high, when compared to strict entropy coding. At the same time, it should be noted that the degree of compression is highly dependent on the data semantic. In general, source encoding may operate either in a lossless or lossy mode.

Source encoding techniques can be classified into three basic types; viz., *transform encoding*, *differential encoding*, and *vector quantization*. Each one of these methods are effective for a signal (or raw data) with certain characteristics. The methods and where they are applicable are explained in this section. Fig. 3.3 provides the bird's eye view of coding techniques.

3.5.1 Transform Encoding

In transform encoding, the *raw data* undergoes a mathematical transformation from the original form in spatial or temporal domain into an abstract domain, which is more suitable for compression. The transform process is a reversible process and the original signal can be obtained by applying the *inverse* transform. Fourier transform and Cosine transform are two popular examples. They transform the signal from the space or time domain to frequency domain.

The important point to note here is that the choice of the type of transformation depends on the type of data. For instance, in the case of images, transforming the signal from the initial time domain to the frequency domain has advantages. The reason is that the spectral representation (i.e., frequency spectrum in the frequency domain) of images captures the changes in color or luminance rapidly. When an image signal $f(x)$ (in spatial domain)

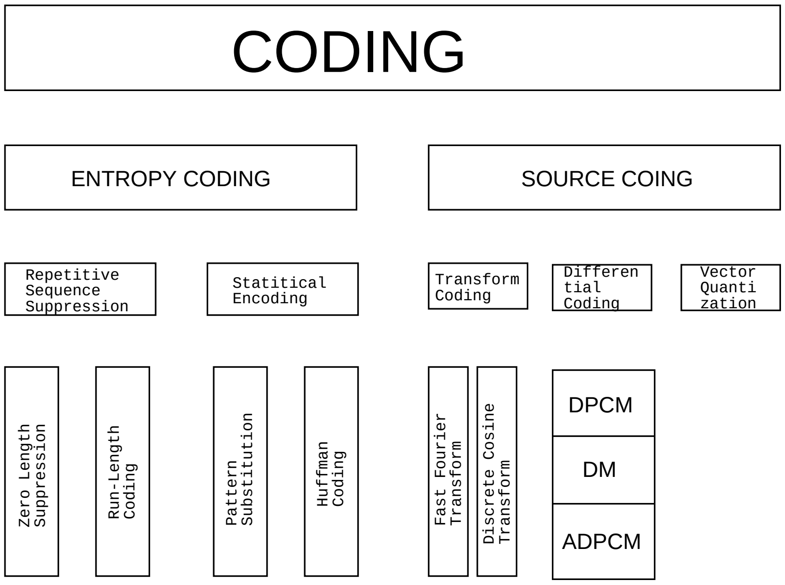


Fig. 3.3. Bird's Eye View of Coding Techniques

is transformed to $F(u)$ (in frequency domain), the DC component and the low frequency components *carry most of the information contained* in the original signal $f(x)$. Even though the signal is an infinite series in the transformed domain, the *most significant* coefficients are *only* in the first few terms. In the quantization stage, normally the less significant coefficients are dropped. Only the *first k coefficients* are actually used. The choice of k, however, depends on the application requirement. Moreover, the significant coefficients can be coded with better accuracy than the less significant ones.

Discrete Cosine Transform (DCT) is the transform encoding technique used in image compression standards such as JPEG. The coefficients (significant part of the transformed signal) retained are called *DCT coefficients*. This method is further discussed in Section 3.8.

3.5.2 Differential Encoding

In differential encoding, only the difference between the actual value of a sample and a prediction of that value is encoded. Because of this approach, differential encoding is also known as predictive encoding. Techniques such as *differential pulse code modulation*, *delta modulation*, and *adaptive pulse code modulation* belong to this class of differential encoding techniques. All

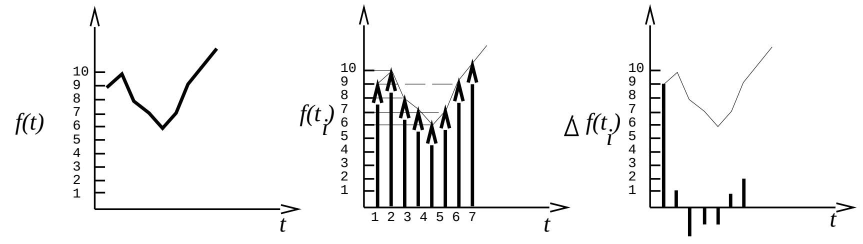


Fig. 3.4. Differential PCM Example

these methods essentially differ in the prediction part.

The differential encoding technique is well suited to signals in which successive samples do not differ much from each other, but they should be significantly different from zero values! These techniques naturally apply to motion video (where one can transmit the difference in images across successive frames) and audio signals.

Differential Pulse Code Modulation (DPCM): In DPCM, the prediction function is simply the following:

$f_{predicted}(t_i) = f_{actual}(t_{i-1})$

So what needs to be encoded at every sampling instant is:

$\Delta f(t_i) = f_{actual}(t_i)$ - $f_{actual}(t_{i-1})$

If successive sample values are close to each other, then only the first sample needs a larger number of bits to encode and the rest can be encoded with a relatively shorter number of bits.

Example: Consider the signal and its samples as shown in Fig. 3.4. The $f_{actual}(t_i)$ at various instances t_1, t_2, t_3, t_4, t_5, t_6, and t_7 are 9, 10, 8, 7, 6, 7, and 9. The $f_{predicted}(t_i)$ are 0, 9, 10, 8, 7, 6, and 7. Therefore, the $\Delta f(t_i)$ are +9, +1, -2, -1, -1, +1, and +2.

Delta Modulation: Delta modulation is a special case of DPCM. The prediction function is the same as DPCM. The difference is in coding the error (difference between the predicted and the actual values). Delta Modulation codes the error as a single bit or digit. It simply indicates that the 'current' sample is to be increased by a step or decreased by a step. Delta modulation is more suited to the coding of signals that do not change too rapidly with the sampling rate.

Adaptive DPCM: ADPCM is the sophisticated version of DPCM. The predicted value is extrapolated from a series of values of preceding samples using a time-varying function. That means, instead of using a constant prediction function, the estimation is made variable to be in line with the characteristics of the sampled signal.

Vector Quantization: In vector quantization, the given data stream is divided into blocks called *vectors*. These blocks can be one- or two-dimensional. Typically, when we deal with images, the blocks are usually q square block of pixels. Often, the vectors are of the same size. Consider the image presented in Fig. 3.5 and its digital representation with a resolution of 8 x 8. In the figure, there are 2 x 2 blocks that together make up the 8 x 8. Each one of the 2 x 2 blocks, will have a pattern between all 0s and all 1s, both inclusive. A table called *code book* is used to find a match for each one of the 2 x 2 blocks, which will correspond to some or all the patterns. Each entry in the code book (basically, it is a table of entries as shown in Fig. 3.5) is a 2 x 2 pattern of 0s and 1s written in a linear form by stacking the rows of the 2 x 2 matrix one after another. This is indicated as *position index* in Fig. 3.5. The code book itself may be predefined or dynamically constructed.

Using the code book, each vector in the image is coded using the best match available. As the code book is present, both at the coding and the decoding ends, only the reference to the entry is actually transferred. When there is no exact match available in the code book, the vector of the image is coded with reference to the *nearest* pattern and will appear as *distortion* in the decoded picture. Also, while coding and transmitting, one can also transfer the errors (difference between the entries in the code book and the vectors in the image) in a quantized form. Depending on the quantization level of the errors and whether or not the errors are sent along with the image, the scheme may become lossy or lossless.

Vector quantization is useful in coding (or compressing) a source whose characteristics are particularly well known. Moreover, it should be possible to construct the code book that can approximate a wide range of actual image vectors. Vector quantization is generally found to be useful in coding of speech. The reader may be interested in understanding more about optimal construction of code books and the algorithms to find out the best pattern match.

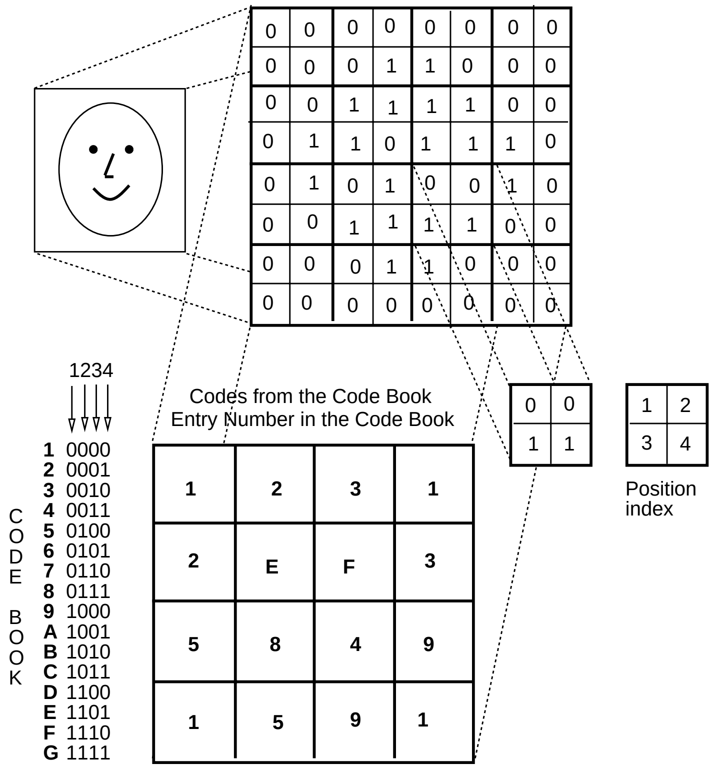


Fig. 3.5. Vector Quantization

3.6 Image Compression System

The uncompressed picture is in analog form and the compressed picture is in digital form and there are several steps involved in compressing and decompressing a picture. The source image data (which is in digital form) is compressed using an encoder and reconstructed from the compressed form, using a decoder. The *encoder* and the *decoder* are the two functional blocks of an image compression system. This is shown in Fig. 3.6. Both the encoder

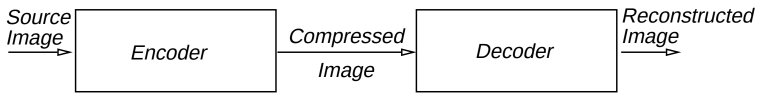


Fig. 3.6. Image Compression System

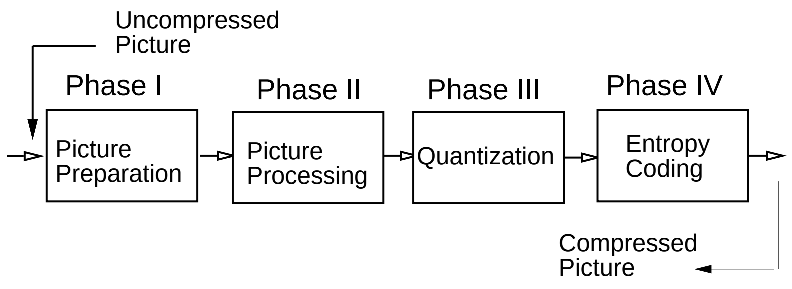


Fig. 3.7. Basic Units of an Encoder

and the decoder have different parts A generic comparison of units that together make up an encoder is shown in Fig. 3.7. These units correspond to different phases in the encoding process. The different phases together convert the analog form of the picture to a corresponding compressed digital stream at the output.

Phase I is the picture preparation phase. In this phase the uncompressed analog signal is converted to its digital counterpart through *sampling.* Each sample is then represented in digital form using the appropriate number of bits per sample. A picture is considered as consisting of a number of pixels in the X and Y axes to cover the entire screen. The picture (pixels) is also divided into *blocks* consisting of 8×8 array of pixels each. This is true for the JPEG [2] standard for still pictures or images[4]. In the case of motion video compression based on MPEG [3], motion-compensation units called Macroblocks of size 16×16 are used. This size is arrived at based on a trade-off between the coding gain provided by motion information and the cost associated with coding the motion information [5,7]. These two standards are discussed further in this chapter.

Phase II is the picture processing phase. In this phase, the actual compression takes place using a source coding or lossy algorithm. A transformation from the time domain to the frequency domain is performed. As most of the 'information' content is in the DC and low frequencies, appropriate weights are used in selecting the coefficients for inter-frame coding in this phase.

Phase III is the quantization phase. In this phase, the output of the previous stage, which are coefficients expressed as real numbers, are mapped into

[2] Joint Photographers Experts Group.
[3] Motion Pictures Experts Group.

integers, resulting in a reduction of precision. In the transformed domain, the coefficients are distinguished according to their significance; for example, they could be quantized using a different number of bits per coefficient.

Phase IV is the entropy encoding phase. In this phase, which is the last step in the compression process, the digital data stream which is output from the quantization phase is compressed without losses.

In some cases, depending on the content of the image, the repeated application of processing phase and quantization phase, will result in better compression. Some compression techniques realize this and gainfully employs them. Such techniques are called adaptive compression schemes, and they essentially repeat Phases II and III several times for better compression. After compression, the data looks as shown in Fig. 3.8. A preamble, the coding technique used, actual data, and possibly an error correction code are all part of a typical compressed picture.

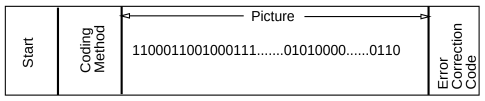


Fig. 3.8. The Compressed Picture

At the receiving end, this information has to be *decompressed* before presentation. Decompression is the inverse of the compression process. If an application uses similar techniques resulting in same execution time for both compression and decompression, then we call that application *symmetric*; otherwise it is called *asymmetric*. Interactive applications are symmetric and presentation applications are asymmetric. Teleconferencing is an example of a symmetric application and an audio-visual tutoring program is an example of an asymmetric application. As we mentioned earlier, depending on the *picture quality* and *time constraints*, appropriate compression technique will be selected. The standards applicable for pictures provide for this option.

3.7 Compression System Standards

There are several coding and compression standards already available; some of them are in use in today's products while others are still being developed. The most important standards are:

- JPEG, standard developed jointly by ISO and ITU-TS for the compression of still images.

- MPEG 1, MPEG 2 and MPEG 4 standards developed by ISO committee IEC/JTC1/SC29/WG11 for coding combined video and audio information.

- H.261, standard developed by Study Group XV and known popularly as Video Coded for Audiovisual Services at $px64$ Kbps.

- ITU-TS H.263 for videophone applications at a bit-rate below 64 Kbps.

- ISO JBIG for compressing bilevel images.

- DVI, a de facto standard for compression from Intel that enables storage compression and real-time decompression for presentation purposes. This will not be discussed further as this standard is getting obsolete.

3.8 JPEG

JPEG is the acronym for Joint Photographic Experts Group, and is a joint collaboration between the ISO committee designated JTC1/SC2/WG10 and the CCITT SGVIII (ITU-T) to establish an international standard for continuous tone (multilevel) still images, both grey scale and color. JPEG can compress typical images from 1/10 to 1/50 of their uncompressed bit size without visibly affecting image quality. JPEG addresses several applications such as photovideotex, color facsimile, medical imaging, desktop publishing, graphic arts, newspaper wire photo transmission, and so on. In 1992, JPEG became an ISO standard and an international standard [4,11,12].

The goal of JPEG has been to develop a method for continuous tone image compression which meets the following requirements:

- Encoder should be parameterizable so that the application can set the desired compression quality tradeoff. The compression scheme should result in image fidelity.

- To be applicable to practically any kind of continuous tone digital source image; not to be restricted to certain dimensions, color spaces, pixel aspect ratio, etc. Not to be limited to classes of scene imagery with restrictions on scene content such as complexity, range of colors, etc.

- Should have tractable computational complexity, to make it possible to implement on a range of CPUs, as well as special-purpose hardware. The compression process has to be completed in real-time.

- Should be possible to implement in hardware with a viable cost for application requiring high performance

- Should have the following modes of operation:

  - *Sequential encoding:* each image is encoded in multiple scans for applications in which transmission time is long and the viewer prefers to watch the image build up in multiple coarse to clear phases.
  - *Lossless encoding:* the image is encoded to guarantee exact recovery of every source image sample value
  - *Hierarchical encoding:* the image is encoded at multiple resolutions so that the lower resolution versions may be accessed without first having to decompress the image at its full resolution

JPEG aims at a very general scheme which is picture-feature independent and technique independent. Basically, it defines a framework that can take care of all the requirements listed above. The stages of JPEG encoding are as shown in Fig. 3.9, the corresponding stages in decoding, decompression and reconstruction of the picture are shown in Fig. 3.10. The JPEG standard is general and is independent of picture size, color, and aspect ratio. JPEG considers a picture as consisting of a number of components, each with a different size. For example, brightness and color are two components of a picture.

An Aside: The *elements* specified in the JPEG standards are *encoder*, *decoder,* and an *interchange format.*

An *encoder* is an embodiment of an encoding process. An encoder takes as input *digital source image data* and *table specifications*, and by means of a specified set of *procedures* generates as output *compressed image data.*

A *decoder* is an embodiment of the *decoding process.* A decoder takes as input compressed image data and table specifications (same as the ones used by the encoder), and by means of a set of procedures generates as output *digital reconstructed image data.*

An *interchange format* is a compressed image data representation which includes all table specifications used in the encoding process. the interchange format is for exchange between application environments.

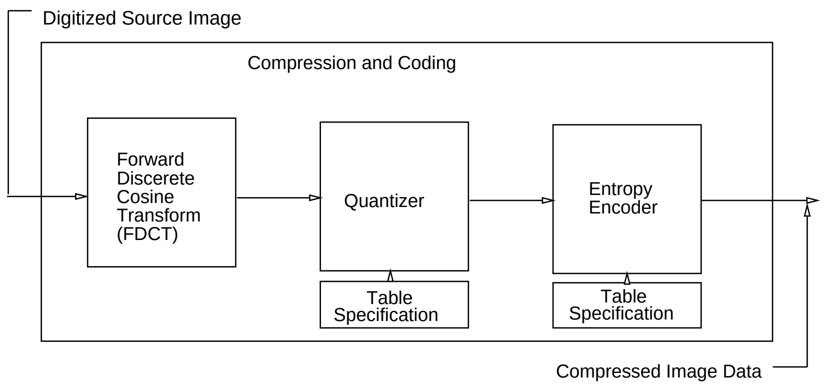


Fig. 3.9. JPEG Encoding of a Picture

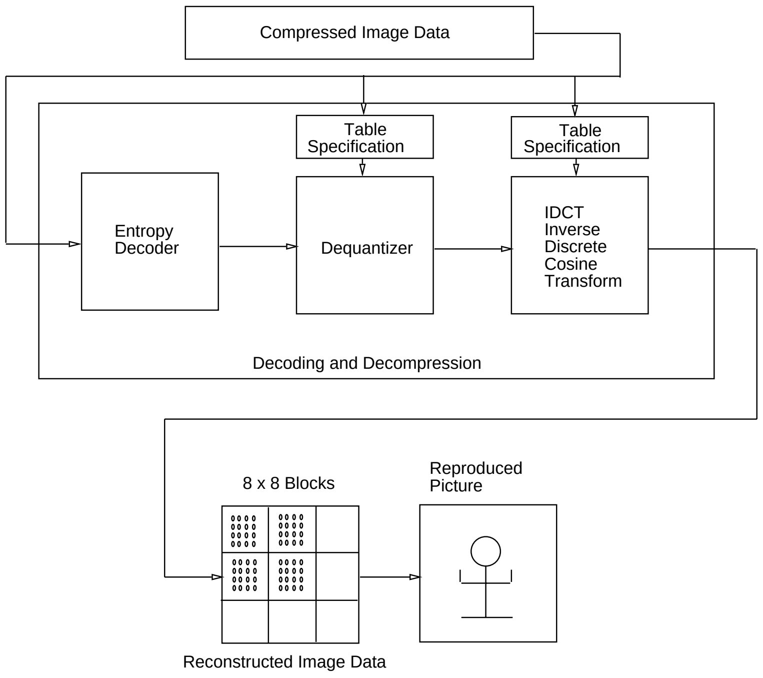


Fig. 3.10. JPEG Decoding of a Picture

3.8.1 JPEG Compression Process

The JPEG standard enables the design of two processes; viz.; image *compression* and *decompression*. In a sequential mode, each one of these processes

- JPEG Compression Process
  - Preparation of Data Blocks
  - Source Encoding step
    - Discrete Cosine Transform(DCT), also called forward DCT(FDCT)
    - Quantization
  - Entropy Encoding step
    - Run Length Coding
    - Huffman or Arithmetic Coding

- JPEG Decompression Process
  - Entropy Decoding Step
    - Huffman or Arithmetic Coding
    - Run Length Coding
  - Source Decoding Step
    - Dequantization
    - Inverse Discrete Cosine Transform(IDCT)

Table 3.1. Steps in the JPEG Process

consists of several steps as shown in Table 3.1.

JPEG may compress grey-scale and color images. As there are many ways to represent color images, the standard does not impose an initial format for the representation of color images. In general, the pictures are considered as a matrix of colored dots called pixels. Each pixel may be represented by an RGB [4] triplet, a YUV (European TV) or YIQ (North American and Japanese TV), a YC_rC_b triplet, or by many other combinations. The digitized variables which represent the image are called the *image components*. R,Y,Q or C_b are examples of image components, which are represented as matrices of values. Sometimes, these components may be sub-sampled - which is normally done in the case of color difference components. As the size of the image to be compressed is variable and the sub-sampling unknown, JPEG has foreseen the need to deal with a relatively large number of image components. The maximum number of components that JPEG can deal with is 255. In practice, regular color images use three components

[4]Composite signals - RGB are direct and the rest are derivatives, as explained earlier in Chapter 2.

only.

JPEG utilizes a methodology based on Discrete Cosine Transform (DCT) for compression. It is a symmetrical process with the same complexity for coding and for decoding. The reader should note that JPEG does not have an embedded encoded/compressed audio signal and is aimed at pictures or images only.

In DCT based coding, two distinct modes are possible: *sequential* and *progressive*. For sequential DCT-based mode, 8x8 sample blocks are typically input block-by-block from left-to-right, and then block-row by block-row top-to-bottom. After a block has been quantized and prepared for entropy encoding, all 64 of its quantized DCT coefficients can be immediately entropy encoded and output as a part of the compressed image data thereby minimizing the storage requirements. In this approach, the image will slowly unfold from top-to-bottom.

For the progressive DCT-based mode, 8x8 blocks are also typically encoded in the same order, but in multiple scans through the image. This is accomplished by adding an image-sized coefficient memory buffer between the quantizer and the entropy encoder. As each block is quantized, its coefficients are stored in the buffer. The DCT coefficients in the buffer are then partially encoded in each of multiple scans. This approach will make a silhouette of the image appear first and then progressively sharpen and brighten it.

Preparation of Data Blocks: As explained earlier in Chapter 2, the source image has to be converted from the analog to digital form. The source digital image in its original form is a matrix of sampled values of amplitude of signals (color or gray scale). If it is color, then there will be as many matrices as there are color components and if it is gray scale, there will be a single component. This is illustrated in Fig. 3.11. The discrete cosine transform is not applied on the entire image. Instead, in the sequential lossy mode, the image is divided into individual blocks. Each individual block is treated separately at each step of the processing chain. The blocks are small square matrices of 8×8 sampled values.

Example: Consider a color image represented by three components: the luminance Y and the two color differences U and V. The size of the image is 640 pixels per line by 480 lines, which is equivalent to computing a VGA standard. Therefore, the luminance consists of a matrix of 640 by 480 values, and each color difference component is a matrix of 320 by 240 values if we assume a 4:1:1 chrominance reduction (Fig. 3.11 and Fig. 3.9). The block preparation step will submit to the DCT process 4800 blocks of luminance and twice 1200 blocks for the color difference. Blocks may then be passed to the DCT, component after component and within each component left to right and top to bottom. This technique is called non-interleaved data ordering.

Forward Discrete Cosine Transform: In transform coding, the initial spatial or temporal domain is transformed into an abstract domain which is more suitable for compression. The process is reversible; that is, applying the inverse transformation will restore the original data.

Each 8×8 block in Fig. 3.9 can be represented by 64 point values denoted as:

$$f(x,y), 0 \leq x \leq 7, 0 \leq y \leq 7, \text{ where x and y are the two spatial domains.}$$

That is, each 8x8 block of source image samples can be viewed as a 64 point discrete signal that is a function of the two spatial dimensions x and y. The DCT transforms these values to the frequency domain as $c = g(F_u, F_v)$ where c is the coefficient and F_u and F_v are the respective spatial frequencies for each direction using the transformation[4]:

$$\text{For } 0 \leq u \leq 7, 0 \leq v \leq 7 \; F(u,v) =$$
$$0.25 * C(u) * C(v) * \sum_{x=0}^{7} \sum_{y=0}^{7} f(x,y) * \cos(((2x+1) * u * \pi)/16) * \cos(((2y+1) * v * \pi)/16),$$
$$\text{where } C(u) = C(v) = 1/\sqrt{2} \text{ for u,v = 0 and 1 otherwise.}$$

The output of this equation gives another set of 64 values *known as the DCT coefficients*, that is the value of a particular frequency - no longer the amplitude of the signal at the sampled position (x,y). The coefficient corresponding to vector (0,0) is called the DC coefficient and the rest are called AC coefficients. The DC coefficient generally contains a significant fraction of the total image energy. Because sample values typically vary slowly from point to point across an image, the FDCT (Forward Discrete

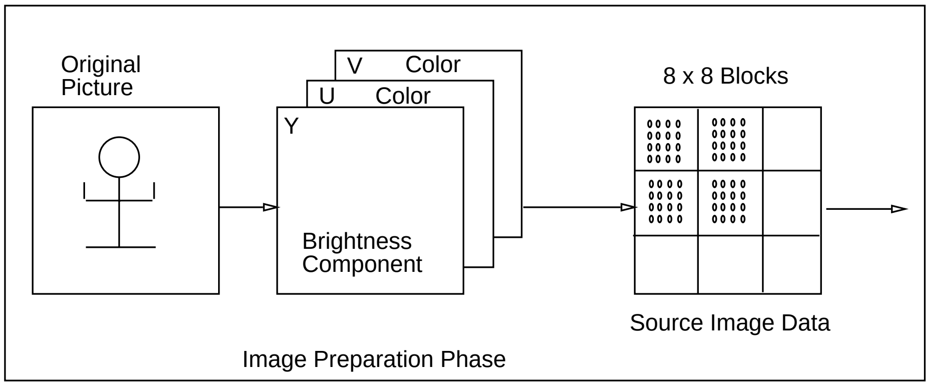


Fig. 3.11. Image Preparation Phase

Cosine Transform) processing compresses data by concentrating most of the signal in the lower values of the *(u,v)* space. For a typical 8x8 sample block from a typical source image, many, if not most, of the *(u,v)* pairs have zero or near-zero coefficients and therefore need not be encoded. At the decoder, IDCT (Inverse Discrete Cosine Transform) reverses this processing step. For each DCT-based mode of operation, the JPEG proposal specifies different codes for 8-bit and 12-bit source-image samples.

Intuition: In a block representing an image, sampled values usually vary slightly from point to point. Thus, the coefficients of the lowest frequencies will be high, but the medium and high frequencies will have a small or zero value. The energy of the signal is concentrated in the lowest spatial frequencies.

Imagine a flat monochromatic wall. Take a picture of it and divide it into 8×8 blocks. These are the *f(x,y)*. In each block, the amplitude of the signal will be nearly constant from pixel to pixel. As the value does not change much in each direction, the zero frequency will be high - as the *frequency measures the rate of change in each direction* - whereas the other frequencies will be nearly zero.

If a sharp black line is drawn on the picture, the image becomes more complex. Some of the blocks will be affected - some not all. In the affected blocks, there will be a rapid change between two consecutive values, which will entail a high coefficient for one of the highest frequencies.

In practice, continuous-tone images such as photographs do not have too many sharp lines or zeros. The transitions between areas are often smooth. Thus, the information is usually mainly contained in the low frequencies. In fact, this is the underlying assumption in JPEG. JPEG is not designed for too complex images, in particular images which look like bitonal images.

In principle, DCT introduces no loss to the source image samples; it simply transforms them to a domain in which they can be more efficiently encoded. This means that, if the FDCT and IDCT could be computed with perfect accuracy and if DCT coefficients were not quantized, the original 8x8 block could be recovered exactly. But as can be seen from the equations, the FDCT (also the IDCT) equations contain transcendental functions. Therefore, no finite time implementation can compute them with perfect accuracy. Several algorithms have been proposed to compute these values approximately. But, no single algorithm is optimal for all implementations: an algorithm that runs optimally in software, normally does not operate optimally in firmware (e.g., in a programmable DSP) or in hardware.

Because of the finite precision of DCT inputs and outputs, coefficients calculated by different algorithms or by independent implementations of the same algorithm will result in slightly different output for identical input. To enable innovation and customization, JPEG has chosen not to specify a unique FDCT/IDCT algorithm, but to address the quality issue by specifying an accuracy test to ensure against largely inaccurate coefficients that degrade image quality.

Quantization: The output from the FDCT (the 64 DCT coefficients) is uniformly quantified in conjunction with a 64 element quantization table to be specified by the application (or the user) as an input to the encoder. Each element can be an integer from 1 to 255, which specifies the step size of the quantifier for its corresponding DCT coefficient. The purpose of quantization is to achieve further compression by representing DCT coefficients with no greater precision than is necessary to achieve the desired image quality. It may be noted that *quantization* is a many-to-one mapping, and therefore is *fundamentally lossy*. It is the principal cause of lossiness in DCT-based encoders. Quantization is defined as division of each DCT coefficient by its corresponding quantifier step size, followed by rounding to the nearest integer:

$$F^Q(u,v) = IntegerRound\frac{F(u,v)}{Q(u,v)}$$

After quantization, the DC coefficient is treated separately from the 63 AC coefficients. The DC coefficient is a measure of the average value of the 64 image samples. Because there is usually a strong correlation between the DC coefficients of adjacent 8x8 blocks, the quantized DC coefficient is encoded as the difference from the DC term of the previous block in the encoding order, as shown in Fig. 3.12. This special treatment is worthwhile, as DC coefficients frequently contain a significant fraction of the total image energy.

Finally, all of the quantified coefficients are ordered into the "zig-zag sequence" as shown in Fig. 3.12. The ordering helps to facilitate entropy coding by placing low-frequency coefficients (which are more likely to be non-zero) before high-frequency coefficients.

The final step in DCT-based encoder is entropy coding. This step achieves additional compression losslessly by encoding the quantified DCT coefficients more compactly based on their statistical characteristics. The JPEG proposal specifies two entropy coding methods - Huffman coding and arithmetic coding. It is useful to consider entropy coding as a two-step process. The first step converts the zig-zag sequence of quantified coefficients into an intermediate sequence of symbols. The second step converts the symbols to a data stream in which the symbols no longer have identifiable boundaries. The form and definition of the intermediate symbols is dependent on both

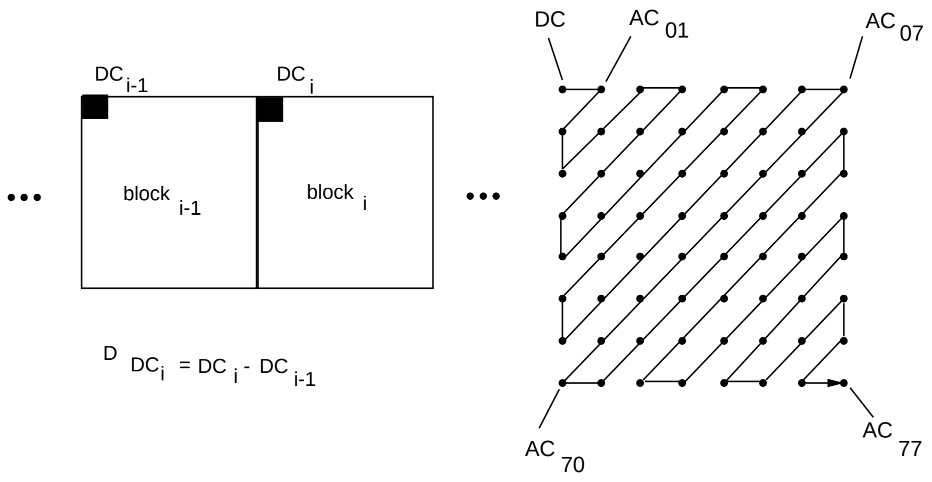


Fig. 3.12. Differential DC Coding and Zig-Zag Sequencing

the DCT-based mode of operation and the entropy coding method. Huffman coding requires that one or more sets of Huffman code tables be specified by the application. The same tables used to compress the image are needed to decompress the image. Huffman tables may be pre-defined and used within an application as defaults, or computed specifically for a given image in an initial statistics-gathering pass prior to compression. Such choices are the responsibility of JPEG applications. As such, the JPEG proposal does not endorse any Huffman tables. A comprehensive example is given in Fig. 3.13. The reader should note that the numbers given in the example are only indicative of what various matrices will look like and not from any real image.

3.8.2 Compression and Picture Quality

For color images with moderate complex scenes, all DCT-based modes of operation typically produce the levels of picture quality for the indicated ranges of compression as shown in Table. 3.2.

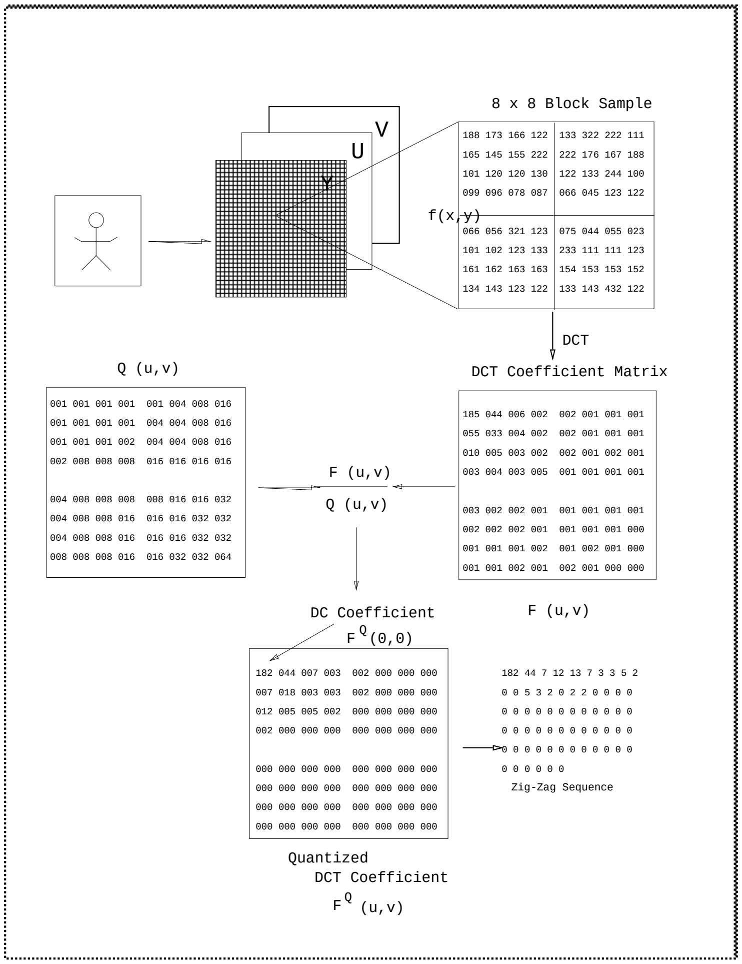
3.8.3 Processing at the Receiver

At the receiving end, upon decoding, de-quantization (which is the inverse function of quantization) is to be performed. That simply means that the normalization is removed by multiplying by step size, which returns the result to representation appropriate for input to the IDCT. The corresponding equation is:

$$F^{Q'}(u,v) = F^{Q}(u,v) * Q(u,v)$$

This is then fed to the IDCT that is represented by the following idealized mathematical definition for a 8x8 block:

$$f(x,y) =$$
$$0.25 * \sum_{x=0}^{7} \sum_{y=0}^{7} C(u) * C(v) * F(u,v) * \cos(((2x+1) * u * \pi)/16) * \cos(((2y+1) * v * \pi)/16)$$
where $C(u) = C(v) = 1/\sqrt{2}$ for u,v = 0 and 1 otherwise.



8 x 8 Block Sample

188	173	166	122	133	322	222	111
165	145	155	222	222	176	167	188
101	120	120	130	122	133	244	100
099	096	078	087	066	045	123	122
066	056	321	123	075	044	055	023
101	102	123	133	233	111	111	123
161	162	163	163	154	153	153	152
134	143	123	122	133	143	432	122

DCT Coefficient Matrix

185	044	006	002	002	001	001	001
055	033	004	002	002	001	001	001
010	005	003	002	002	001	002	001
003	004	003	005	001	001	001	001
003	002	002	001	001	001	001	001
002	002	002	001	001	001	001	000
001	001	001	002	001	002	001	000
001	001	002	001	002	001	000	000

Q (u,v)

001	001	001	001	001	004	008	016
001	001	001	001	004	004	008	016
001	001	001	002	004	004	008	016
002	008	008	008	016	016	016	016
004	008	008	008	008	016	016	032
004	008	008	016	016	016	032	032
004	008	008	016	016	016	032	032
008	008	008	016	016	032	032	064

Quantized DCT Coefficient

182	044	007	003	002	000	000	000
007	018	003	003	002	000	000	000
012	005	005	002	000	000	000	000
002	000	000	000	000	000	000	000
000	000	000	000	000	000	000	000
000	000	000	000	000	000	000	000
000	000	000	000	000	000	000	000
000	000	000	000	000	000	000	000

```
182 44 7 12 13 7 3 3 5 2
0 0 5 3 2 0 2 2 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0
```

Fig. 3.13. JPEG Example

Intuitively, FDCT and IDCT behave like harmonic analyzer and harmonic synthesizer, respectively. At the decoder, the IDCT takes the 64 DCT coefficients and reconstructs a 64-point output image signal by summing the basis signals.

3.8.4 Hierarchical Mode of Operation

In hierarchical mode of operation, an image is encoded as a sequence of *frames*. These frames provide *reference reconstructed components* which are usually needed for prediction in subsequent frames. Like in the progressive DCT-based mode, hierarchical mode also offers a progressive presentation. It is useful in environments which have multi-resolution requirements. Hierarchical mode also offers the capability of progressive transmission to a final lossless stage.

Using hierarchical coding, a set of successively smaller images can be created by *down-sampling* (low-pass filtering and sub-sampling) the preceding larger image in the set. Then starting with the smallest image in the set, the image set is coded with increasing resolution. After the first stage, each lower-resolution image is scaled up to the next resolution (up-sampled) and used as a prediction for the following stage. When the set of images is stacked (as shown in Fig. 3.14), it sometimes resembles a pyramid. Just as smaller images generated as part of a hierarchical progression can be scaled up to full resolution, full-resolution images generated as part of a non-hierarchical

Compression Range	Picture Quality
0.25-0.5 bits/pixel	Moderate to good quality, sufficient for some applications
0.5-.075 bits/pixel	Good to very good quality, sufficient for many applications
0.75-1.5 bit/pixel	Excellent Quality, sufficient for most applications
1.5-2.0 bits/pixel	Usually indistinguishable from the original picture, sufficient for the most demanding applications

Table 3.2. Compression Range versus Picture Quality [4]

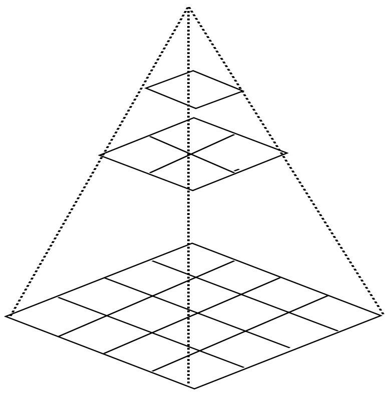


Fig. 3.14. Hierarchical Multi-Resolution Encoding

progression can be scaled down to smaller sizes. These scalings are done at the decoder and are not restricted to powers of 2.

3.8.5 Multiple-Component Images

In the JPEG standard, a source image may contain 1 to 255 image components, also referred to as color or spectral bands or channels. Each component consists of a rectangular array of samples. All samples are defined to be an unsigned integer with precision P bits, with any value in the range [0, 2^p-1]. All samples of all components within the same source image must have the same precision P, the value of which can be 8 or 12 for DCT-based codes and 2 to 16 for predictive codes, respectively. The multicomponent source image model is shown in Fig. 3.15.

The component C_i has sample dimensions $x_i \times y_i$. As the formats in which some image components are sampled may differ (compared to other components), in the JPEG source model, each component can have different dimensions. The dimensions must have a mutual integral relationship defined by H_i and V_i, the relative horizontal and vertical sampling factors, which must be specified for each component. Overall, image dimensions X and Y are defined as the maximum x_i and y_i for all components in the image, and can be any number up to 2^{16}. H and V are allowed only the integer values 1 through 4. The encoded parameters are X, Y, and the H_is and V_is for each component. The decoder reconstructs the dimensions x_i and y_i for each component, according to the following ceiling function:

$$x_i = \text{X} * H_i / H_{max}$$

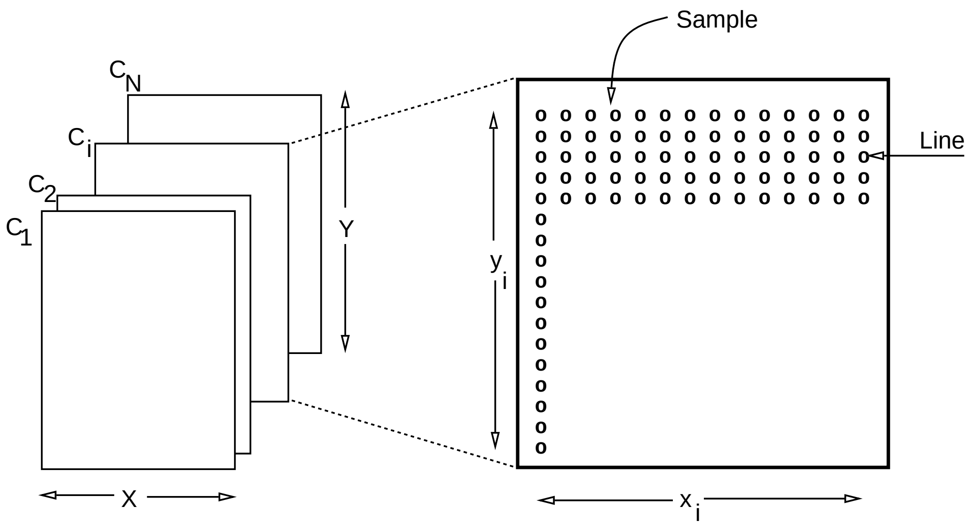


Fig. 3.15. JPEG Source Image Model

$$y_i = \mathrm{Y} * V_i/V_{max}$$

Example: In the example discussed so far, the C_is are Y, U, and V, the corresponding H_i and V_i are $(H_Y = 4, V_Y = 4), (H_u = 2, V_U = 2)$, and $(H_V = 2, V_V = 2)$ for Y, U, and V, respectively. The resolution of each component Y, U, and V can be assumed to be $(X_Y = 512, Y_Y = 512), (X_U = 256, Y_U = 512)$, and $(X_V = 256, Y_V = 512)$.

3.8.6 Encoding Order and Interleaving for Multi-Component Images

In practice, many applications need to pipeline the process of displaying multi-component images in parallel with the process of decompression. For many systems, this is only feasible if the components are interleaved together within the compressed data stream. The JPEG proposal defines the concept of a *data unit*, which is a single sample in the case of predictive codecs and an 8x8 block of samples in the case of DCT-based codecs. When two or more components are interleaved, each component C_i is partitioned into rectangular regions of x_i by y_i data units, as shown in Fig. 3.15. Regions are ordered within a component from left-to-right and top-to-bottom, and within a region, data units are ordered from left-to-right and top-to-bottom. The JPEG proposal defines the term Minimum Coded Unit *(MCU)* to be the

	C_1						C_2						C_3			C_4		
	0	1	2	3	4	5	0	1	2	3	4	5	0	1	2	0	1	2
0	A	B	E	F	I	J	a	b	c	d	e	f	0	2	4	*	+	=
1	C	D	G	H	K	L	g	h	i	j	k	l	1	3	5	#	$	%
2	M	N	Q	R	U	V	m	n	o	p	q	r	6	8	0	@	!	&
3	O	P	S	T	W	X	s	t	u	v	w	x	7	9	1	^	-	)
4	Y	Z																
5																		

MCU$_1$ = ABCD ab 01 *

MCU$_2$ = EFGH cd 23 +

MCU$_3$ = IJKL ef 45 =

MCU$_4$ = MNOPgh 67 #

Fig. 3.16. Interleaved Data Ordering Example

smallest group of interleaved data units. In the example given in Fig. 3.16, MCU_1 consists of data units taken from the top-left most region of C_i, followed by data units from the same region of C_2, and likewise for C_3 and C_4. MCU_2 continues the pattern as shown in Fig. 3.16. Thus the interleaved data is an ordered sequence of *MCUs,* and the number of data units contained in an *MCU* is determined by the number of components interleaved and their relative sampling factors. The maximum number of components which can be interleaved is 4, and the maximum number of data units in an *MCU* is 10.

3.8.7 Using Multiple Tables

When multiple components and interleaving are involved, JPEG codes must control the use of a proper data table for each component. It should be ensured that the same quantization table with the same entropy coding table

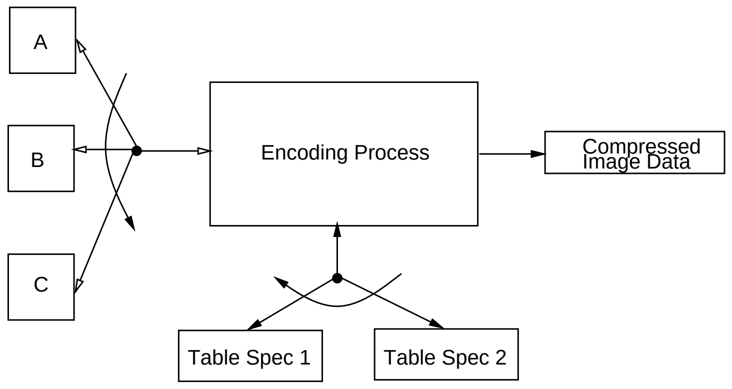


Fig. 3.17. Table Switching Control

be used to encode all samples within a component. JPEG decoders can store up to four different quantization tables and the same entropy coding table must be used to encode all samples within a component. Therefore, it is necessary to switch between tables during decompression of a scan containing multiple interleaved components, as the tables cannot be loaded during decompression of a scan. Fig. 3.17 illustrates the task switching control that must be managed in conjunction with multiple component interleaving for the encoder side.

3.8.8 Some Remarks on JPEG

Most importantly, an interchange format syntax is specified which ensures that a JPEG compressed image can be exchanged successfully between different application environments. The applications have the freedom to specify default or referenced table, as appropriate.

In essence, JPEG is a standard to be strictly followed for Image Exchange and efficient hardware, software, and hybrid implementations are available [14]. While a multimedia system designer cannot afford to change the JPEG process, the designer can certainly fine-tune the implementation.

3.9 JBIG

Images can be broadly classified into two categories: *bi-tonal* and *continuous-tone* images. A regular printed text without any gray-scale characters is an ideal example of a bi-tonal image. JBIG concentrates on bi-tonal images while JPEG concentrates on continuous-tone images. Bi-tonal images are also called *bi-level* images (that is, an image that has only colors like black and white). ITU-T Recommendations T.4 and T.30 are coding standards for Group 3 facsimile machines. Compression algorithm of the Group 3 standard was lain down in 1990 for use with switched telephone networks. The coding standard recommendation T.6 for Group 4 facsimile machines was published in 1984. While T.6 provides better resolution, it requires line speeds of the order of 56 Kbps or 64 Kbps to operate.

JBIG (Joint Bi-level Image Group) is an advanced compression scheme utilizing lossless, predictive methods [6,8,9]. The JBIG compression algorithm is defined by ISO/IEC Standard 11544:1993. The JBIG specification defines the compression scheme, not the file format. The JBIG Alliance is sponsoring the JBIG Interchange Recommendation, which provides a well-defined method for storing JBIG-compressed data in industry standard TIFF (Tagged Image File Format)-formated files.

JBIG is an advanced lossless compression algorithm originally developed for facsimile transmission. The efficiency of the algorithm derives from arithmetic encoding, a mathematically complex concept based on image contexts. Because JBIG stores data in independent bit-planes, JBIG efficiently compresses both bi-tonal and gray-scale data.

The main characteristics of JBIG are:

- Compatible progressive/sequential coding. This means that a progressively coded image can be decoded sequentially, and the other way around.

- JBIG will be a lossless image compression standard: all bits in the images before and after compression and decompression will be exactly the same.

JBIG specifies the number of bits per pixel in the image. Its allowable range is 1 through 255, but starting at 8 or so, compression will be more efficient using other algorithms. On the other hand, medical images such as chest X-rays are often stored with 12 bits per pixel while no distortion is allowed, so JBIG can certainly be of use in this area. To limit the number of bit changes between adjacent decimal values (e.g., 127 and 128), it is wise to use Gray coding before compressing multi-level images with JBIG. JBIG then compresses the image on a bit-plane basis, so the rest of this text assumes bi-level pixels.

Progressive coding is a way to send an image gradually to a receiver instead of all at once. JBIG uses discrete steps of detail by successively doubling the resolution. Compatibility between progressive and sequential coding is achieved by dividing an image into stripes. Each stripe is a horizontal bar with a user-definable height. Each stripe is separately coded and transmitted, and the user can define in which order stripes, resolutions, and bit-planes are intermixed in the coded data. A progressive coded image can be decoded sequentially by decoding each stripe, beginning by the one at the top of the image, to its full resolution, and then proceeding to the next stripe. Progressive decoding can be done by decoding only a specific resolution layer from all stripes.

After dividing an image into bit-planes, resolution layers, and stripes, eventually a number of small bi-level bitmaps are left to compress. Compression is done using a Q-coder.

The Q-coder codes bi-level pixels as symbols using the probability of occurrence of these symbols in a certain context. JBIG defines two kinds of context: one for the lowest resolution layer (the base layer), and one for all other layers (differential layers). Differential layer contexts contain pixels in the layer to be coded, and in the corresponding lower resolution layer.

For each combination of pixel values in a context, the probability distribution of black and white pixels can be different. In an all-white context, the probability of coding a white pixel will be much greater than that of coding a black pixel. The Q-coder assigns, just like a Huffman coder, more bits to less probable symbols, and so achieves compression. The Q-coder can, unlike a Huffman coder, assign one output code-bit to more than one input symbol, and thus is able to compress bi-level pixels without explicit clustering, as would be necessary using a Huffman coder.

Maximum compression will be achieved when all probabilities (one set for each combination of pixel values in the context) follow the probabilities of the pixels. The Q-coder therefore continuously adapts these probabilities to the symbols it sees.

3.10 MPEG

The MPEG system layer has the basic task of combining one or more audio and video compressed bit-streams into a *single bit-stream.* It defines the data stream syntax that provides for timing control and the interleaving and synchronization of audio and video bit-streams. From the systems perspective, MPEG bit-stream consists of a *system layer* and *compression layers.* The system layer basically provides an envelope for the compression layers. The compression layers contain the data to be given to the decoders, whereas the system layer provides the control for *demultiplexing* the interleaved compression layers. A typical MPEG system block diagram is shown in Fig. 3.18.

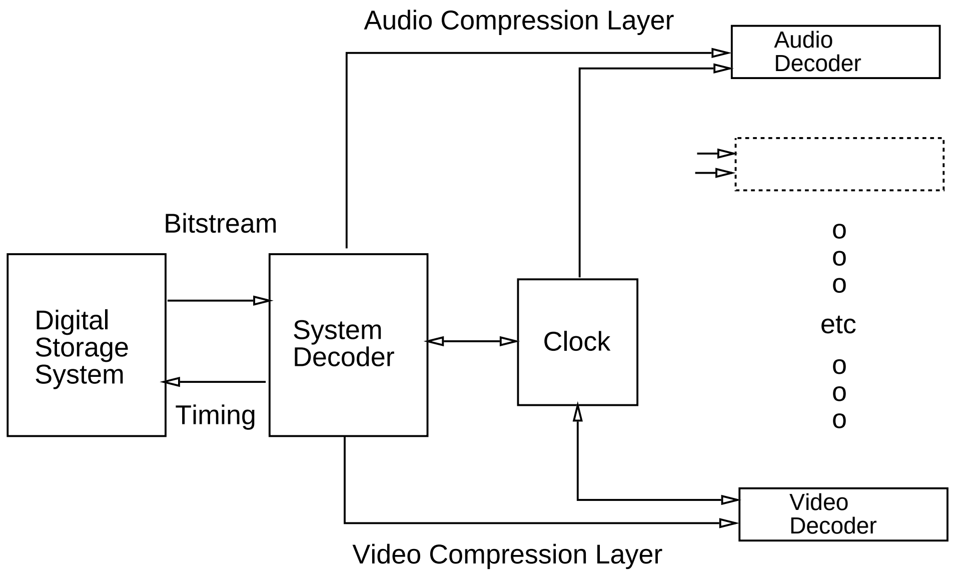


Fig. 3.18. MPEG System Structure

The MPEG bit-stream consists of a sequence of packs that are in turn subdivided into packets, as shown in Fig. 3.22. Each pack consists of a unique 32-bit byte aligned pack start code and header, followed by one or more packets of data. Each packet consists of a packet start code (another unique 32-bit byte aligned code) and header, followed by a packet data (compressed audio or video data). The system decoder parses this bit-stream and feeds the separated audio and video to the appropriate decoders along with timing information.

The MPEG system uses an idealized decoder called System Target Decoder (STD), which interprets the pack and packet headers to deliver the elementary bit-streams to the appropriate audio or video decoder. In STD the bits for an access unit (a picture or an audio access unit) are removed from the buffer instantaneously at a time dictated by a Decoding Time Stamp (DTS) in the bit-stream. The bit-stream also contains another type of time stamp, the Presentation Time Stamp (PTS). Buffer overflow or underflow is controlled by the DTS; synchronization between audio and video decoding is controlled by the PTS.

3.10.1 MPEG Audio Compression

MPEG audio coding supports 32, 44.1, and 48 KHz. At 16 bits per sample the uncompressed audio would require about 1.5 Mbps. After compression, the bit-rates for monophonic channels are between 32 and 192 Kbps; the bit-rates for stereophonic channels are between 128 and 384 Kbps.

The MPEG audio system first segments the audio into windows of 384 samples wide and uses a filter bank to decompose each window into 32 sub-bands, each with a width of approximately 750 Hz (for a sample of 48 KHz). As is typical of sub-band coding, each sub-band is decimated such that the sampling rate per sub-band is 1.5 KHz and there are 12 samples per window. An FFT (Fast Fourier Transform) of the audio input is used to compute a global masking threshold for each sub-band, and from this a uniform quantizer is chosen that provides the least audible distortion at the required bit-rate. For more information on MPEG audio coding, the reader is referred to [3].

3.10.2 MPEG Video Compression

MPEG is a generic compression standard that addresses video compression, audio compression, and the issue of audio-video synchronization for sequences (movie or VCR output). MPEG compresses the video and audio signals at about 1.5 Mbps per channel. The MPEG system addresses the issue of synchronization and multiplexing of multiple compressed audio and video list streams. The motivation for the MPEG standard comes from the need to handle full-motion video for storage and retrieval with random access. MPEG is designed to compress across multiple frames and therefore can yield compression ratios of 50:1 to 200:1 instead of 20:1 or 25:1 in JPEG. Its algorithm is asymmetrical; that is, it requires more computational complexity (hardware) to compress than to decompress it. This is useful for applications where the signal is produced at one source but is distributed to many, or compressed once and decompressed several times.

At this point, the reader should recall that JPEG deals with a (single) still image (picture) and the compression was an effort to reduce the *spatial redundancy* in the source. In the case of motion video, we are dealing with a stream of (single) still images (pictures) that are available from the source at a constant rate. A sequence of video frames inherently carries a lot of redundant information. Therefore, MPEG compression concentrates on reducing the *temporal redundancy across the frames* in addition to the *spatial redundancy* within a frame. The MPEG standard incorporates a special concept called "group of pictures" (GOP) to deal with temporal redundancy and to allow random access to be stored MPEG-coded videos.

There are three types of pictures that are considered in MPEG; viz.; Intra pictures (I-Frames), predicted pictures (P-Frames) and bi-directional (interpolated) pictures (B-Frames). The I-Frame is the first frame of each *GOP*. The Intra pictures (I-Frames) provide reference points for random access and are subjected to moderate compression. From the compression point of view, I-pictures are equivalent to *images* and can be DCT encoded using a JPEG-like algorithm. In P- and B-pictures, the motion-compensated prediction errors are DCT coded. Only forward prediction is used in the P-pictures, which are always encoded relative to the preceding I- or P-pictures. The prediction of the B-pictures can be forward, backward, or bidirectional relative to other I- or P-pictures. In addition to these three, there is a fourth type of picture called D-pictures, containing only the DC component of each block. They are mainly useful in browsing at very low bit-rates. The number of I, P, and B frames in a *GOP* is application dependent. For example, it depends on the access time requirement and the bit-rate requirement.

Example: In Fig. 3.19 the relationship between the different pictures and what their relative roles are is shown. The example introduces an intra-coded picture every 8 frames and the sequence of I, B, P is IBBBPBBBI. Note that the first frame is an I-frame. In MPEG, the order in which the pictures are processed is not necessarily the same as their time sequential order. The pictures in the example can be coded as 1,5,2,3,4,9,6,7,8 or 1,2,3,5,4,9,6,7,8, since the the prediction for P- and B-pictures should be based on pictures that are already transmitted.

Prediction pictures are coded with reference to a past picture (intra(I) or predicted(P)) and can, in general, be used as a reference for future predicted pictures. Interpolated pictures or B-pictures achieve the highest amount of compression and require both a past and future reference frame. In addition, bi-directional pictures (B-Pictures) are never used as a reference. In all cases, when a picture is coded with respect to a reference, *motion compensation* [5] is used to improve the coding efficiency. Fig. 3.20 expands on the example and shows how they act as inter-frame coding and the relative sizes of the compression information process. As in the example, an intra coded picture every 8 frames and the sequence of I, B, P is IBBBPBBBI. Motion compensation is the basic philosophy behind the coding of B and P frames.

[5] A technique used to compress video.

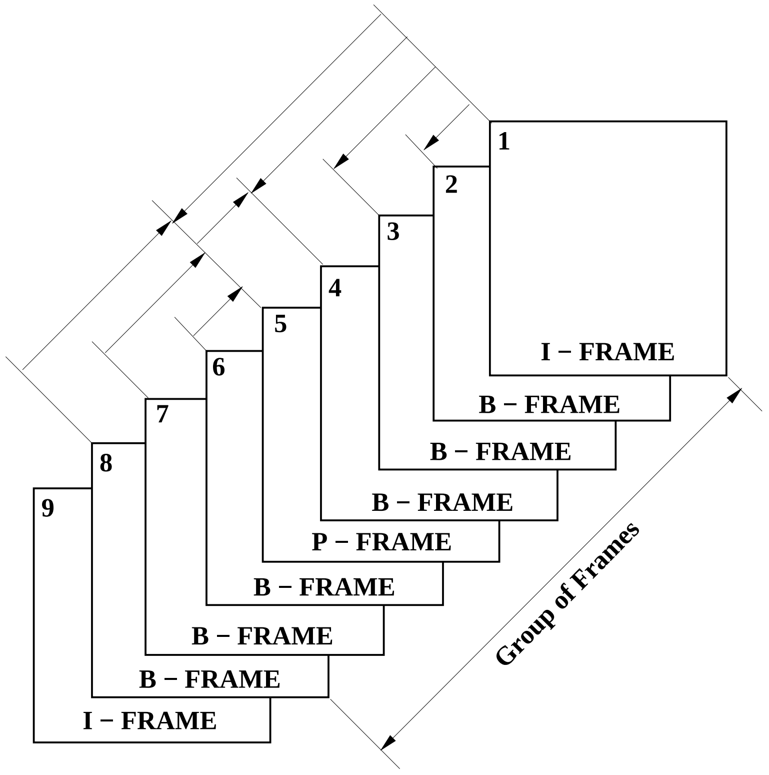


Fig. 3.19. Group of Frames

3.10.3 Motion Compensation

There are two techniques that are employed in motion compensation: prediction and interpolation. Motion-compensated prediction assumes that the current picture can be modeled as a transformation of the picture at some previous time. This means that the signal amplitude and the direction of displacement need not be the same everywhere in the picture. Motion-compensated interpolation is a multi-resolution technique; a sub-signal with low temporal resolution (typically 1/2 to 1/3 the frame rate) is coded and the full resolution signal is obtained by interpolation of the low-resolution signal and addition of a correction term. The signal to be reconstructed by interpolation is obtained by adding a correction term to a combination of a past and future reference [3,5,7].

3.10.4 Motion Representation

MPEG uses a *Macro block* consisting of 16 $\times$ 16 pixels. There is always a trade-off between coding gain provided by the motion information and the cost associated with coding the motion information. The choice of 16 $\times$ 16 blocks for the motion compensation unit is based on such a trade-off. In a bi-directionally coded picture, each 16 $\times$ 16 macro block can be of type Intra, Forward-Predicted, Backward-Predicted, or Average.

3.10.5 Motion Estimation

The extraction of motion information from a video sequence is called motion estimation. If we use the block matching technique, the motion vector is obtained by minimizing a cost function measuring the mismatch between a block and each predictor candidate. The search range V of the possible

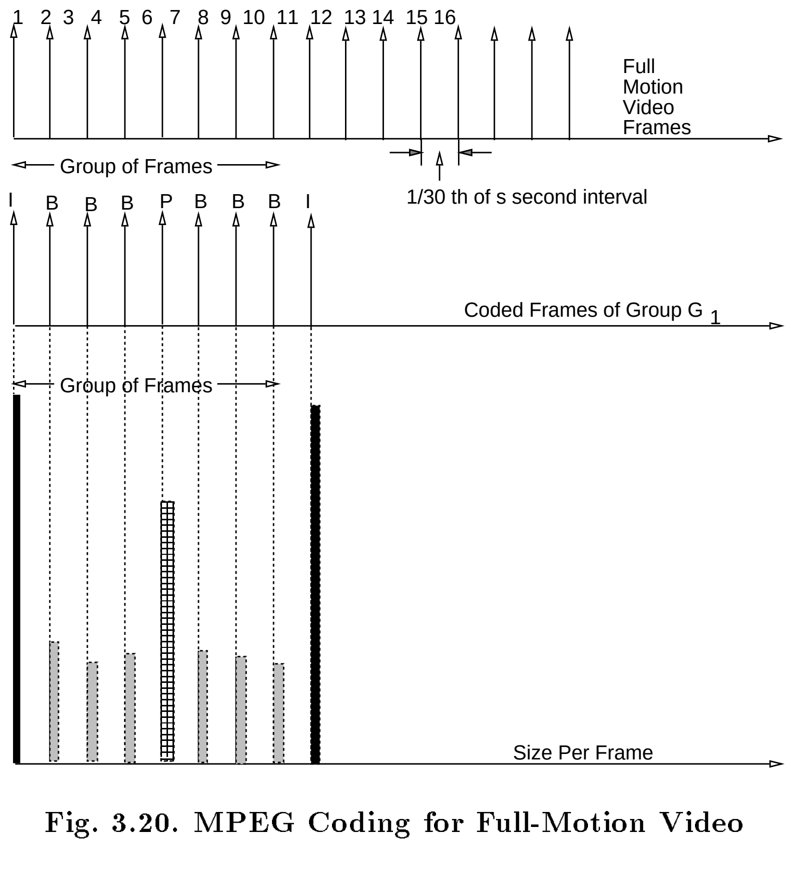


Fig. 3.20. MPEG Coding for Full-Motion Video

Sequence Layer	Random Access Unit: Context
Group of Pictures Layer	Random Access Unit: Video Coding
Picture Layer	Primary Coding Unit
Slice Layer	Re-synchronization Unit
Macro Block Layer	Motion Compensation Unit
Block Layer	DCT Unit

Table 3.3. Six Layers of the MPEG Video

motion vectors and the selection of the cost function are left to the implementation. Exhaustive searches, where all the possible motion vectors are considered, are known to give good results, but at the expense of a very large complexity of computation for large ranges.

3.10.6 Layered Structure, Syntax, and Bit-stream

The MPEG syntax allows for provision of many application-specific features without penalizing other applications. For example, random access and easy accessibility require many access points implying groups of pictures of short duration (e.g., 6 pictures, 1/5 second) and coded with a fixed amount of bits (to make edit-ability possible). Similarly, in order to provide robustness in the case of broadcast over noisy channel, the predictors are frequently reset and each intra and predicted picture is segmented into many slices. The composition of an MPEG video bit-stream contains six layers, as given in Table. 3.3.

Each layer supports a definite function: either a signal processing function (DCT, motion compensation) or a logical function (Re-synchronization, Random Access Point). The MPEG syntax defines a MPEG bit-stream as any sequence of binary digits.

3.10.7 MPEG System

The MPEG standard is a three-part standard; viz., video coding, audio coding, and system coding. The MPEG system coding part specifies the system multiplexer and encoder to produce a MPEG stream from the encoded video and audio streams with a system reference clock as the base.

That is, the MPEG stream is the synchronization of elementary streams and carries enough information in the data fields to do the following tasks:

- Parsing the multiplexed stream after a random access.
- Managing coded information buffers in the decoders.
- Identifying the absolute time of the coded information.

The most important task of the system coding process is the actual multiplexing. It includes the coordination of input data streams and output data streams, the adjustment of clocks, and buffer management. The system's semantic rules impose some requirements on the decoders and allow freedom in the implementation of the encoding process. Fig. 3.21 shows the schematic of such an encoder at the functional level. The video encoder receives the uncoded digitized picture in units called Video Presentation Units (VPUs) at discrete time intervals; similarly, at discrete time intervals, the audio encoder receives uncoded digitized blocks of audio samples in units called Audio Presentation Units (APUs). The video and audio encoders encode digital video and audio, producing coded pictures called Video Access Units (VAUs) and Audio Access units (AAUs). These outputs are referred to as elementary streams. The system encoder and multiplexer produces a multiplexed stream that contains the elementary streams and some additional information to help in handling synchronization and timing. MPEG supports audio from 32 Kbps to 384 Kbps in single channel or in two stereo channels.

Timing and Synchronization: The audio and video sample rates at the encoder are significantly different from one another, and may or may not have an exact and fixed relationship to one another. Also, the duration of a block of audio samples or APUs is generally not the same as the duration of a video picture. There is a single, common system clock in the encoder, and this clock is used to create time stamps that indicate the correct presentation and decoding timing of audio and video, as well as to create time stamps that indicate the instantaneous values of the system clock itself at sampled intervals. The time stamps that indicate the presentation time of audio and video are called Presentation Time Stamps (PTS); those that indicate the decoding time are called Decoding Time Stamps (DTS); and those that indicate the value of the system clock are called the System Clock Reference (SCR). It is the presence of the common system clock in the encoder, the

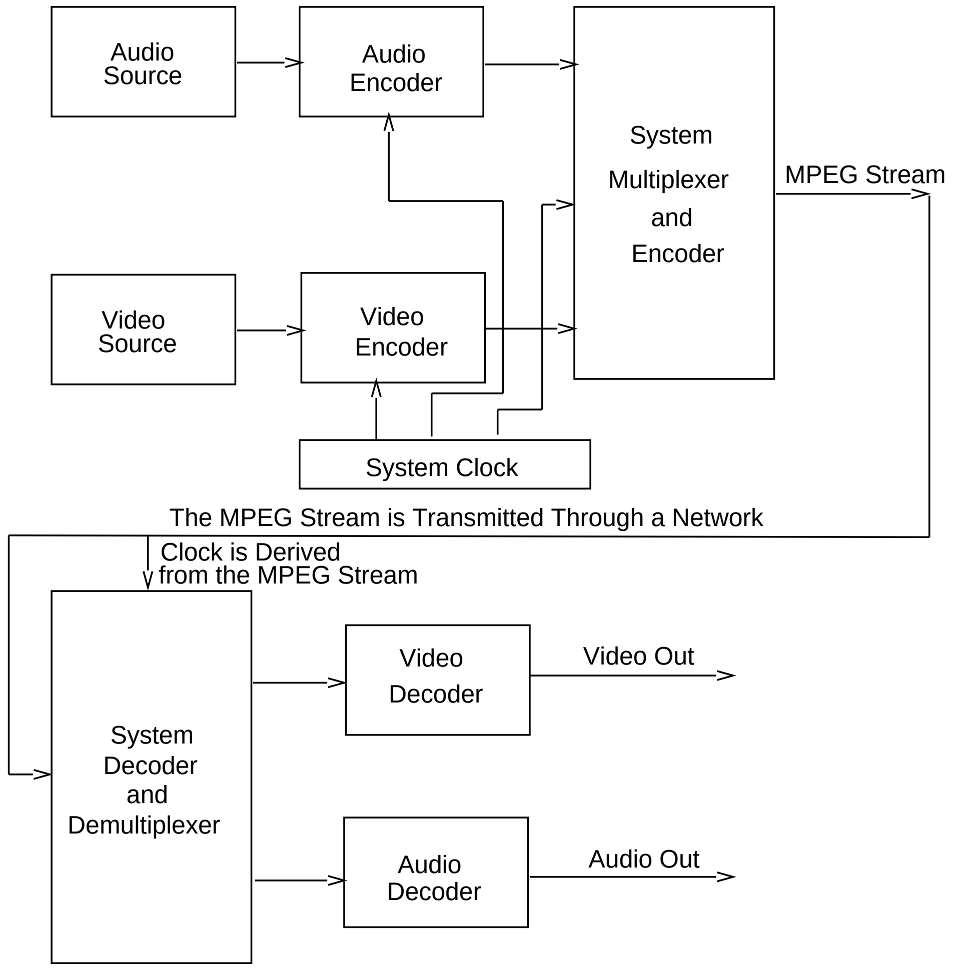

Audio Source
Audio Encoder
System Multiplexer and Encoder
MPEG Stream
Video Source
Video Encoder
System Clock
The MPEG Stream is Transmitted Through a Network
Clock is Derived from the MPEG Stream
System Decoder and Demultiplexer
Video Decoder
Video Out
Audio Decoder
Audio Out

Fig. 3.21. MPEG Data Stream Generation Model

time stamps that are created from it, the recreation of the clock in the decoder, and the correct use of the time stamps that provide the facility to synchronize properly the operation of the decoder.

The MPEG system embodies a timing model in which all digitized pictures and audio samples that enter the encoder are presented exactly once each, after a constant end to end delay, at the output of the decoder. As such, the sample rates, i.e., the video picture rate and the audio sample rate, are precisely the same at the decoder as they are at the encoder. Constant delay, as envisaged in the MPEG system, is required for correct synchro-

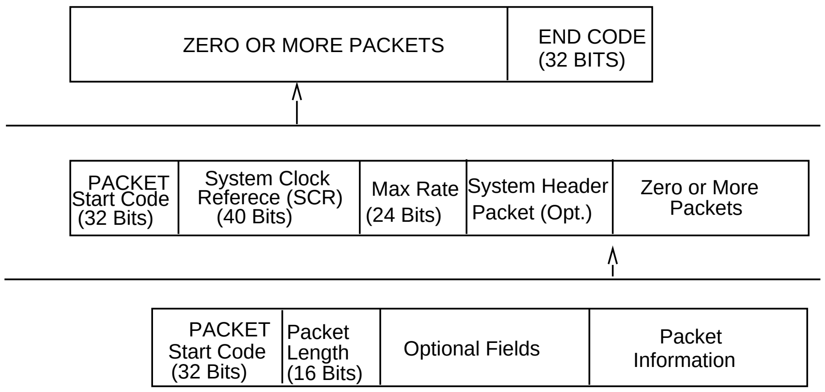


Fig. 3.22. MPEG Bit-stream Layering

nization; however, some deviations are possible due to the variable delays at the encoding buffer during the transmission through the network and at the decoding buffer.

To achieve synchronization in multimedia systems that decode multiple video and audio signals originating from a storage or transmission medium, there must be a 'time master' in the decoding system. MPEG does not specify which entity is the time master. The time master can be any of the decoders, the information stream source, or an external time base. All other entities in the system (decoders and information sources) must slave their timing to the time master. For example, if a decoder is taken as the time master, the time when it presents a presentation unit is considered to be the correct time for the other entities. If the information stream source is the time master, the SCR values indicate the correct time at the moment these values are received. The decoders then use this information to pace their decoding and presentation timing.

MPEG Stream: The most important task of MPEG stream generation process is multiplexing. It includes a combination of input data streams and output data streams, the adjustment of clocks, and the buffer management. The MPEG stream syntax consists of three coding layers: stream layer, pack layer, and packet layer, as shown schematically in Fig. 3.22. Of these three layers, it is the packet layer which actually carries the information from the elementary streams; each packet carries information from exactly one elementary stream. Zero or more packets are grouped together to form a pack, which carries the SCR value. The decoder gets the information necessary for its resource reservation from this multiplexed stream. The maximal data rate is included in the first pack at the beginning of each MPEG stream. Though MPEG does not specify directly the method of multiplexing the elementary streams such as VAUs and AAUs, MPEG does lay down some constraints that must be followed by an encoder and multiplexer in order to produce a *valid* MPEG data stream. For example, the individual stream buffers must not overflow or underflow. That is, the sizes of the individual stream buffers impose limits on the behavior of the multiplexer. Decoding the MPEG data stream starting at the beginning of the stream is straightforward since there are no ambiguous bit patterns. But starting the decoding operation at random points requires locating pack or packet start codes within the data stream.

MPEG Decoding Process: The MPEG standard defines the decoding process - not the decoder. There are many ways to implement a decoder, and the standard does not recommend a particular way. The decoder structure given in Fig. 3.23 is a typical decoder structure with a buffer at the input of the decoder. The bit-stream is demultiplexed into overhead information such as motion information, quantizer step-size, macroblock type, and quantized DCT coefficients. The quantized DCT coefficients are dequantized, and are input to the Inverse Discrete Cosine Transform (IDCT). The reconstructed waveform from the IDCT is added to the result of the prediction. Because of the particular nature of the bidirectional prediction, two reference pictures are used to form the predictor.

3.10.8 MPEG-2, MPEG-3, MPEG-4

MPEG standard is an optimal solution for a target data rate of about 1.5 Mbps. The optimality is with respect to perceptual quality and not necessarily performance. To address the high-quality video requirement (with a target rate of about 40 Mbps), MPEG-2 video standard has been developed. MPEG-2 builds upon MPEG standard (also referred to as MPEG-1 standard) and supports interlaced video formats and HDTV. MPEG-3 was originally started for HDTV and, after the development of MPEG-2 which addresses this issue, MPEG-3 has been dropped. MPEG-4 looks at the other end of the spectrum; low bit-rate coding of audiovisual programs. It is expected to be used in mobile telephony systems.

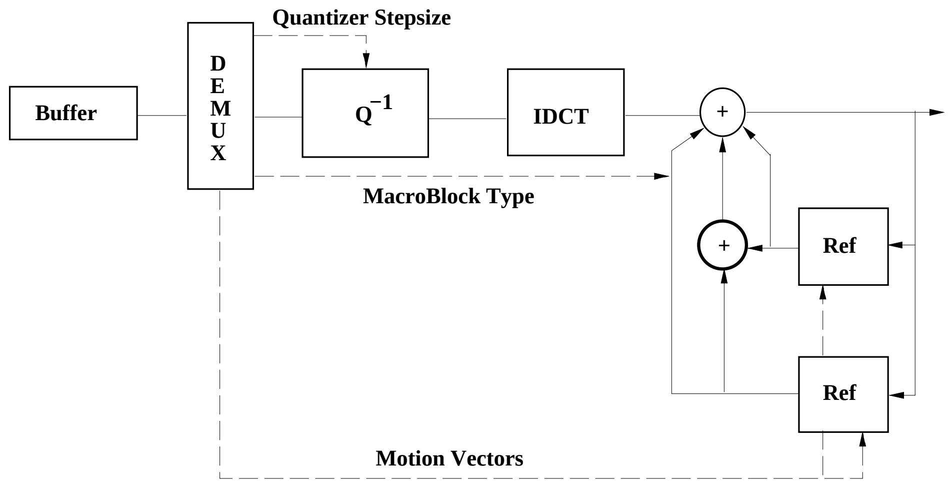


Fig. 3.23. MPEG Decoding Process

3.11 H.261 Standard

CCITT Recommendation H.261 is a video compression standard developed to facilitate videoconferencing and videophone services over the Integrated Services Digital Network (ISDN) at *px64 Kbps, p = 1,...30* [6,13]. For example (when $p=1$), one 64 Kbps may be appropriate for low-quality videophone service, where the video signal can be transmitted at a rate of 48 Kbps and the remaining 16 Kbps is used for audio signal. videoconferencing services generally require higher image quality, which can be achieved with $p \geq 6$, that is, 384 Kbps or higher. Note that the maximum available bit-rate over an ISDN channel is 1.92 MBPS *(p = 30)*, which is sufficient to obtain VHS quality images.

H.261 has two important features:

- It specifies a maximum coding delay of 150 milliseconds, because it is mainly intended for bidirectional video communication. It has been estimated that delays exceeding 150 milliseconds do not give the viewer the impression of direct visual feedback.

- It is amenable to low-cost VLSI implementation, which is important for widespread commercialization.

To permit a single recommendation for use in and between regions using 625 and 525 line TV standards, the H.261 input picture is specified in Common Intermediate Format (CIF). For lower bit-rates a smaller format QCIF, which is one quarter of the CIF, has been adopted. The number of active pels per line has luminance and chrominance are 360 and 180 (352 and 176 visible) for CIF and 180 and 90 (176 and 88 visible) for QCIF, respectively. The number of active lines per picture has luminance and chrominance 288 and 144 for CIF and 144 and 72 for QCIF, respectively. The temporal rates are adjustable for 30, 15, 10, 7.5, for both CIF and QCIF. If we consider 30 frames per second, the raw data rate for CIF will be 37.3 Mbps and for QCIF 9.35, respectively. It may be noted that even with QCIF images at 10 frames per second, 48:1 compression is required for videophone services over a 64 Kbps channel. CIF images may be used when $p \geq 6$, that is, for videoconferencing applications. Methods for converting from CIF to QCIF, or vice versa, is not specified by the recommendation.

3.12 H.263

H.263 is a low bit-rate video standard for teleconferencing applications that has both MPEG-1 and MPEG-2 features [6,7]. It operates at 64 Kbps. The video coding of H.263 is based on that of H.261. In fact, H.263 is an extension of H.261 and describes a hybrid DPCM/DCT video coding method. Both H.261 and H.263 use techniques such as DCT, motion compensation, variable-length coding, and scalar quantization. Both use the concept of macroblock structure.

The idea of PB frame in H.263 is interesting. The PB frame consists of two pictures being coded as one unit. The name PB is derived from the MPEG terminology of P-pictures and B-pictures. Thus, a PB frame consists of one P-picture that is predicted from the last decoded P-picture and one B-picture that is predicted from both the last decoded P-picture and the P-picture currently being decoded. This last picture is called a B-picture because parts of it may be bi-directionally predicted from the past and future P-picture.

It has been shown that the H.263 system typically outperforms H.261 by 2.5 to 1. That means, for a given picture quality, the H.261 bit-rate is approximately 2.5 times that of H.263 codec.

3.13 Summary

In this chapter, we described the image compression system and explained some of the key coding and compression techniques. The concepts relating to coding, compression, and transformation of digital samples of analog signals to their frequency domain equivalents were discussed at length. The two popular coding standards, called JPEG and MPEG, were explained in some detail. JPEG is aimed at still pictures and MPEG is aimed at motion pictures. Other standards, such as H.261, H.263, and JBIG, were discussed to enable the reader to understand the purpose and positioning of different standards. It may be pointed out that this chapter is one of the *key* chapters in the entire book, as it explains the nature of the one dominant media in all multimedia systems, viz., the video. Further information about multimedia systems can be obtained from [10, 11, 12].

REFERENCES

1. C. Shannon A Mathematical Theory of Communication, Bell Systems Technical Journal, Vol. 27, pp. 379-423, July 1948; and pp. 623-656, October 1948.
2. John G. Proakis. Digital Communications, 3rd Edition, McGraw Hill Inc., 1995.
3. D. Pan. An Overview of the MPEG Audio Compression Algorithm, In SPIE. Digital Video Compression on Personal Computers: Algorithms and Technologies, Vol.2187, pp. 260-273, 1994.
4. Gregory K. Wallace. The JPEG Still Picture Compression Standard, Communications of the ACM, Vol.34, No.4, pp.31-44, April 1991.
5. Didier Le Gall. MPEG: A Video Compression Standard for Multimedia Applications, Communications of the ACM, Vol.34, No.4, pp. 46-58, April 1991.
6. Richard Schaphorst. videoconferencing and Video-telephony: Technology and Standards, Artech House Inc., 1996.
7. Joan L. Mitchell, William B. Pennebaker, Chad E. Fogg, and Didier J. Legall (Eds.). MPEG Video Compression Standard, Chapman & Hall, 1997.
8. www.jbig.org
9. ISO/IEC JTC1/SC2/WG9. Progressive Bi-level Image Compression, Revision 4.1, CD 11544, September 16, 1991
10. W.B. Pennebaker, J.L. Mitchell, G.G. Langdon, and R.B. Arps. An Overview of the Basic Principles of the Q-Coder Adaptive Binary Arithmetic Coder, IBM Journal of Research and Development, Vol.32, No.6, November 1988, pp. 771-726.
11. Ralf Steinmetz and Klara Nahrstedt. Multimedia: Computing, Communications and Applications, Prentice-Hall, Inc., 1995.
12. Francois Fluckiger. Understanding Networked Multimedia: Applications and Technology, Prentice-Hall, Inc., 1995.
13. A. Murat Tekalp. Digital Video Processing, Prentice-Hall, Inc., 1995.
14. William B. Pennebaker and Joan L. Mitchell. JPEG : Still Image Data Compression Standard, Van Nostrand Reinhold, 1992.

Exercises

1. Find the digitized values of the samples of the signal given in Fig. 3.24. Each sample is 8 bits.

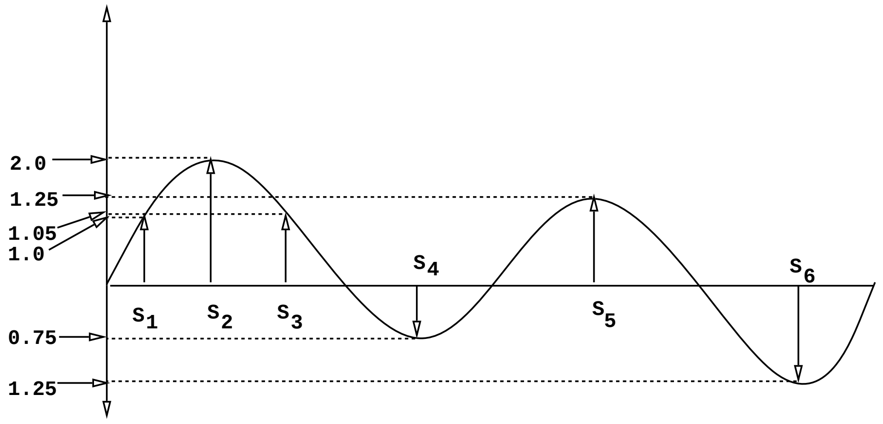


Fig. 3.24. Signal for Exercise 1 in Chapter 3

2. Calculate the Fourier Transform of the digital samples from the previous exercise.
3. An 8x8 pixel formation has elements with values that are inverse of $(r+s)^2$, where r and s are row and column numbers. Compute the value of the pixel matrix.
4. Find the FDCT of the pixel matrix from the previous exercise.
5. The weights to be associated with each FDCT coefficient from the previous exercise are inversely proportional to the position value. Encode the values as per this weightage, keeping the order of the coefficients the same as the order in the pixel matrix (left-to-right and top-to-bottom). Use run-length coding for coding the resulting string.
6. Repeat the previous exercise, using the zig-zag pattern. Use run-length coding for coding the resulting string.
7. Scan your picture after disabling compression in the scanner. Now compress the picture using the JPEG standard. Scan the picture again with compression enabled in the scanner. Compare.
8. Obtain an MPEG Trace from the Internet. Find out the order in which I, P, and B frames appear in that trace.
9. Find out the sizes of the I, P, and B frames from any MPEG trace. Calculate the maximum and minimum size for all three types of frames.