Chapter 17
Compression

17.1 Lossy and Lossless Compression:

In almost all multimedia applications, compression is first applied to the source information prior to transmission. This is necessary to reduce the volume of information to be transmitted and to reduce the bandwidth needed for transmission. A compression algorithm is applied in the encoder at the source, while at the destination, a decompression algorithm is used by the decoder. The compression and decompression can be done in software or hardware. In computer – computer communication compression is done in software since time is not critical, otherwise hardware is used (Fig. 17.1). Lossless (reversible) compression aims at reducing the amount of source information in such a way that when the compressed information is decompressed there is no loss of information. Text files must use lossless compression.

Lossy compression on the other hand aims at reducing the information content by eliminating some of the details that do not affect much the total information. Thus, the interest here is not to reproduce an exact copy of the source information after decompression, but rather a version of it which is perceived by the recipient as close to a true copy as possible. This applies to digitized images, audio and video streams. This is based on the fact that the sensitivity of the human eye or ear can tolerate fine details, hence, such details may be omitted without much noticeable loss of information (Fig. 17.2).

Lossless compression may be divided to entropy - based and source-based. Lossy compression is source based. Entropy-based compression may be divided to statistical encoding and runlength encoding. Statistical encoding uses a set of variable length, and makes use of the probability of occurrence. The shortest codewords are assigned to represent the most frequently occurring symbols. The destination must know the set of codewords being used by the source. This type of encoding is particularly useful with text. One popular type is Huffman, which has the prefix property (Chapter 14). The runlength method considers long substrings of the same character or binary digit. Instead of transmitting the source string in the form of independent codewords or bits, it is transmitted in the form of a different set of codewords indicating the particular character or bit string repeatedly transmitted and the order of its repetition.

The destination must know the set of codewords being used, so it simply interprets each codeword received and outputs the appropriate number of characters or bits . An example is (0001111001111111) is transmitted as (0,3), (1,5), (0,2), (1,7). We may even be content if the first bit is always 0 to transmit 3,5,2,7 in binary form, with the same number of bits per codeword.

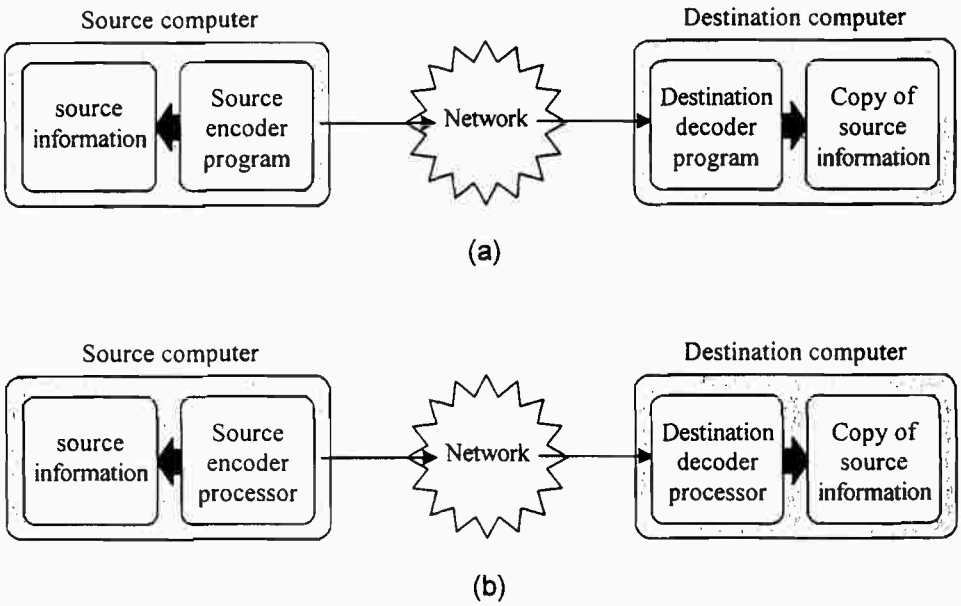


Fig. (17.1) Compression and decompression
a) software b) hardware

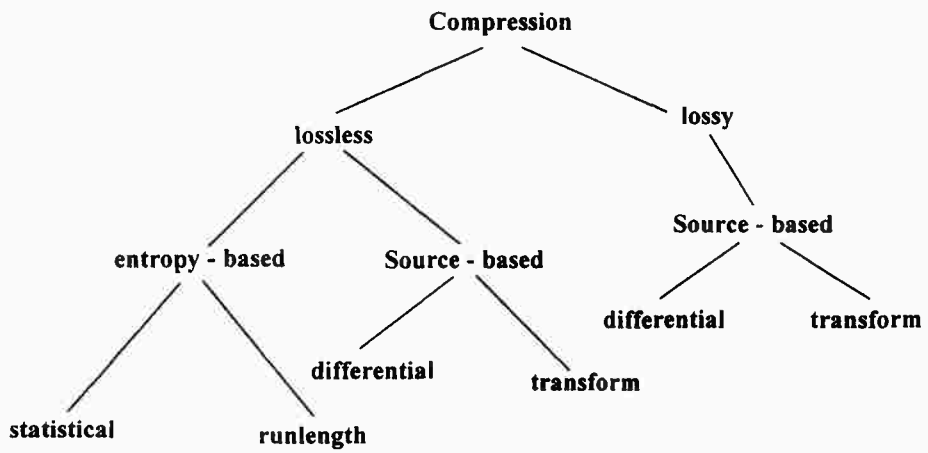


Fig. (17.2) Types of compression

Source-based compression exploits a basic property of the source information to produce only differences with respect to same reference point or frame (differential encoding). It could also use an alternative representation using some transform (transform encoding).

Differential encoding is used extensively in applications, where the amplitude differs only slightly between successive values, while the signal covers a large range. Instead of using a set of long codewords to represent the amplitude of each value - mindless of the uniformity of the signal value over a large region - a set of smaller codewords are used to represent only the differences in amplitude between the present value and the preceding one. Thus, instead of using, e.g., 12 bit, we may use 3 bits (75% saving) in bandwidth. Differential encoding can be either lossless or lossy. If the number of bits used is sufficient to account for the maximum difference, it is lossless. If the difference value exceeds the maximum number of bits being used, some loss of information will result.

In transform coding, the source information is changed from one form to another that is more suitable for compression. It can be lossless or lossy. Digitization of a continuous tone (monochromatic) image gives a 2D matrix of pixel values, each of which represents the level of grey in a particular position of the image. As we go from one position of the matrix to the next, the magnitude of the signal may vary. So as we scan across a set of pixel locations the rate of change of magnitude will vary from zero - if all pixel values are unchanged - to a low rate if half the pixels have one value and the other half have another value, to a high rate if each pixel has a different value. Thus the rate of change of magnitude as we traverse the matrix in the horizontal direction gives a horizontal spatial frequency. Likewise, there is a vertical spatial frequency. Thus, for a particular image, there will be a mix of different spatial horizontal and vertical frequencies. Noting that the human eye is less sensitive to higher frequency components than to lower frequency components, and if the amplitudes of the higher frequency components fall below a certain threshold they will not be detected by the eye, and hence, can be neglected.

Thus, we may transform the original spatial representation into equivalent spatial frequency components, eliminating the high components that the eye does not discern, thus, reducing the content of information and the required bandwidth without degrading the quality of the original image.

The transformation of a 2D matrix of pixel values to an equivalent matrix of spatial frequency components can be carried out using a mathematical technique called discrete cosine transform (DCT). The output is lossless, except for rounding off errors. However, if a threshold is set to get rid of the high frequency components of low values, then this process is lossy, but is good enough for the human eye.

Normally, a graphical image is represented in the form of a program. This takes less memory and bandwidth than the corresponding matrix of pixels. In the case of digitized documents and still pictures, once digitized, the only form of representation is a 2D matrix of pixels. Transforming a graphical image in this program form uses lossless compression. However, if the created image or graphic is to be transformed across the network in its bit map form, then

compression must be applied prior to transmission. To transfer a digitized image compression may employ runlength and statistical coding (lossless compression). It is used for the transfer of digitized documents generated by scanners such as in FAX. For bitonal and color digitized pictures, a combination of transforms, differential and runlength is employed. There are two formats which help in the bit map transmission, graphics interchange format (GIF) and tagged image file format (TIFF). GIF is particularly helpful with the internet. Instead of using 8 bits for RGB (leading to 24 bit depth), only 256 entries are chosen out of the original set of 2^{24} colors that match most closely those in the original image, .i.e., the resulting table of colors consists of 256 entries each of which contains a color code. Instead of assigning each pixel as a 24 bit value, only 8 bit index in the table entry that contains the closest match color to the original is sent.

This achieves 3:1 compression. Contents of the table are sent across the network together with the compressed image data along with other information such as screen size and aspect ratio. TIFF supports pixel resolution up to 48 bits (16 bit for each of R,G,B).

In both cases of GIF and TIFF, a form of compression called Lempel-Ziv Welsh coding is used. Instead of sending the codeword, only the index indicating where it is found in the table is sent. This is called dictionary - based compression algorithm. The dictionary is held by both encoder and decoder, and the contents of the dictionary are built dynamically.

17.2 DCT:

Note that the spatial frequency indicates the rate of change. So in a 10 pixel long matrix, the lowest change is 0 (dc), then 2 means changes in every 5 pixels. The maximum is 10 changes in 10 pixels (Fig. 17.3).

Thus, spatial frequency gives the sampling frequency, which is needed to represent the changes in pixel values. Thus, the square wave has the highest rate of change, and hence, the most extended frequency content. We usually end up with $N \times N$ components to match $N \times N$ pixels.

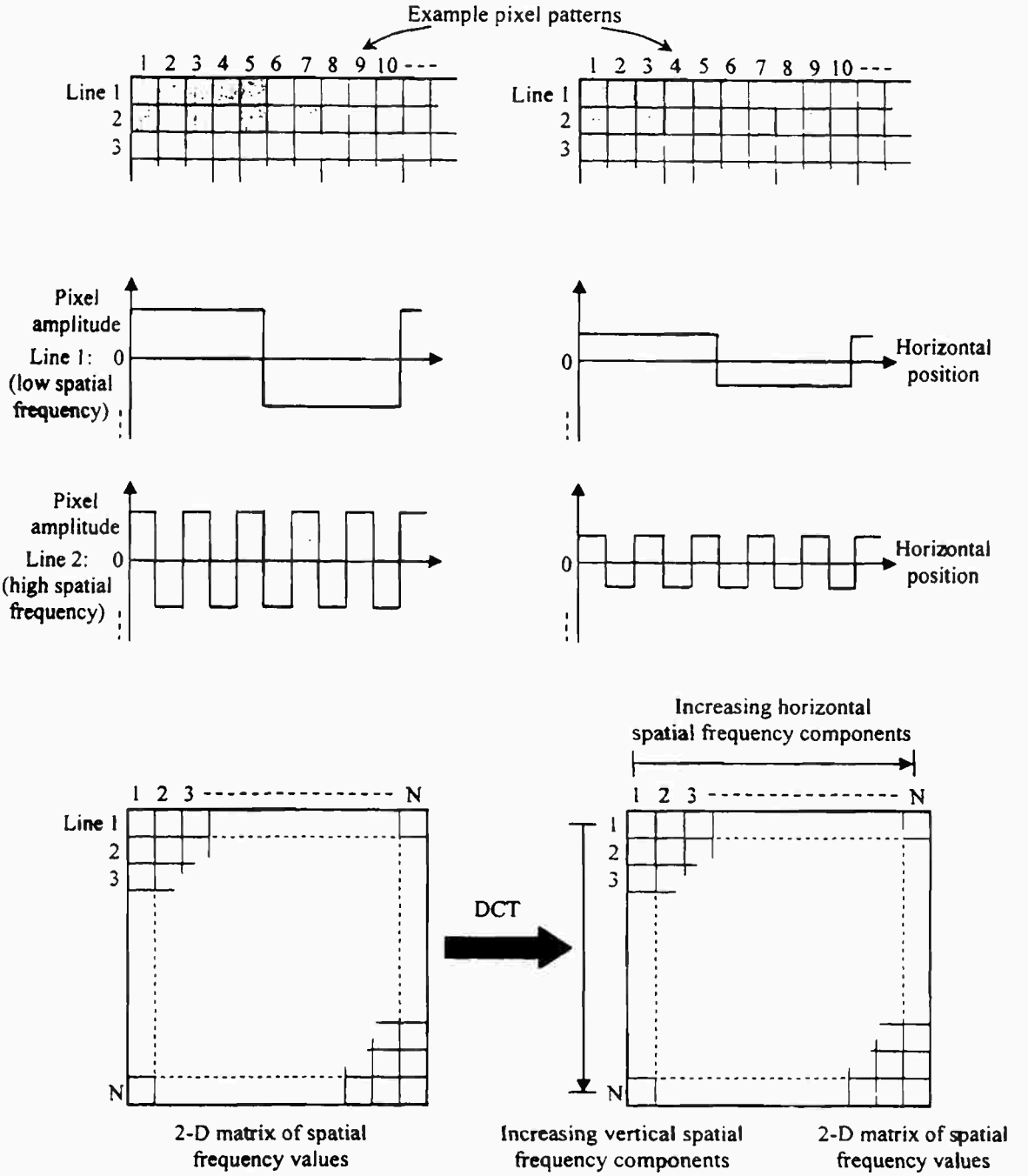
In Fourier transform in 1D

$$F(u) = \int_{-\infty}^{\infty} f(x)\, e^{-j2\pi u x}\, dx \qquad (17-1)$$

In 1 D DCT, we have

$$C(u) = \alpha(u) \sum_{x=0}^{N-1} f(x) \cos\left[\frac{(2x+1)u\pi}{2N}\right]$$

$$u = 0, 1, 2 \ldots N-1 \qquad (17-2)$$



DCT = discrete cosine transform

Fig.17.3 Transform coding

a) pixel patterns *b) DCT transformation*

The inverse Fourier transform is

$$f(x) = \int_{-\infty}^{\infty} F(u)\, e^{j2\pi ux} du \qquad (17-3)$$

The inverse DCT in 1D is given by

$$f(x) = \sum_{u=0}^{N-1} \alpha(u)\, C(u) \cos\frac{(2x+1)\,u\,\pi}{2N} \qquad (17-4)$$

for $x = 0, 1, 2 \ldots N-1$

$$\alpha(u) = \frac{1}{\sqrt{N}} \qquad u = 0 \qquad (17-5)$$

$$\sqrt{2/N} \qquad u = 1, 2, .. N-1$$

For 2D

$$C(u, v) = \alpha(u)\, \alpha(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) \cos\left[\frac{(2x+1)u\pi}{2N}\right] \cos\left[\frac{(2y+1)v\pi}{2N}\right]$$

$$u, v = 0, 1, 2 \ldots N-1 \qquad (17-6)$$

$$f(x, y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} \alpha(u)\, \alpha(v) \cos\left[\frac{(2x+1)u\pi}{2N}\right] \cos\left[\frac{(2y+1)v\pi}{2N}\right]$$

$$x, y = 0, 1, 2 \ldots N-1 \qquad (17-7)$$

Note that for $u = 0$, $v = 0$ $\cos 0 = 1$, giving dc. This is why cosine transform is used not sine. Note also that DCT deals with real quantity unlike Fourier transform.

17.3 JPEG:

This is a compression standard applied to 2D arrays of pixel values. It stands for Joint Photographic Expert Group. It can use different modes. The one described here is called lossy sequential (base line) mode. It consists of the following stages (Fig. 17.4).

1) Image / Block representation.
2) Forward DCT.
3) Quantization.
4) Entropy.
5) Frame building.

In the case of a continuous tone monochromatic image, a single 2D matrix is required to store the set of 8 bit grey level values that represent the image. In the case of color images, instead of using three 2D matrices, a color look up table (CLUT) is used, hence again, a single 2D matrix may be used. However, if the image is represented in an RGB format, 3 matrices are required for RGB or Y, C_b, C_r, noting that C_b and C_r require chrominance of $1/2$ the bandwidth of luminance signal Y. This allows the 2 matrices that contain the digitized chrominance components to be smaller in size than the Y matrix.

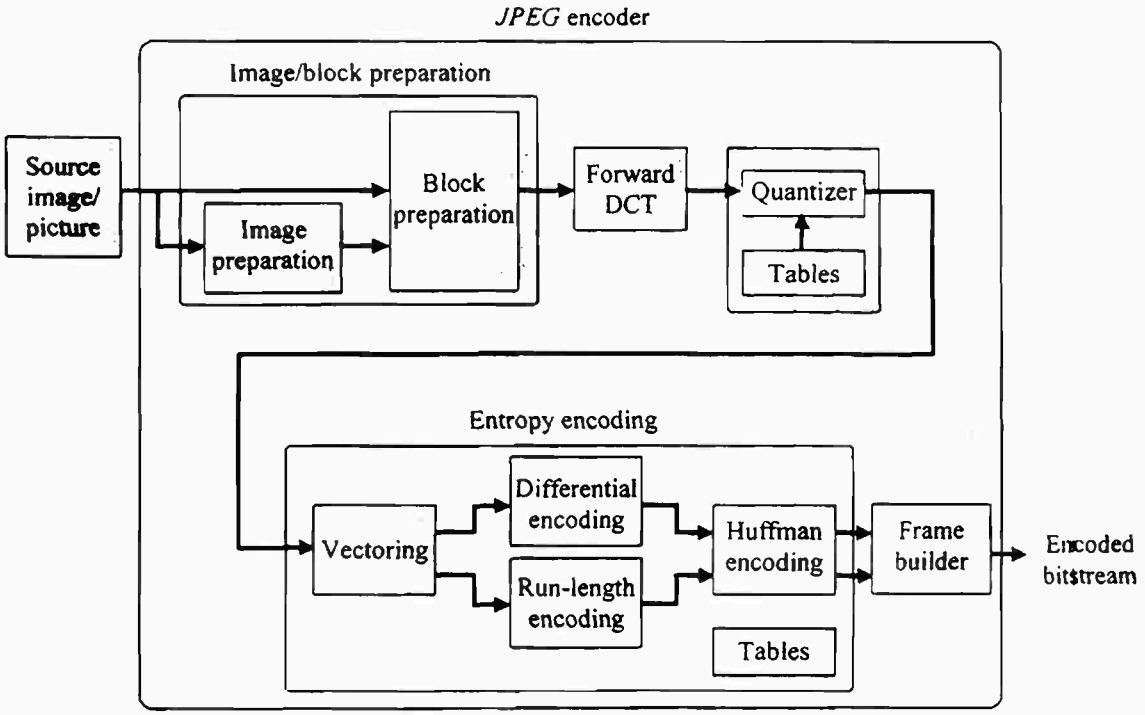


Fig. (17.4) JPEG encoder block diagram

Once the source image format has been selected, the set of values in each matrix are compressed separately, using DCT. Before performing DCT, however, block preparation is carried out. This is necessary to reduce the computational effort in DCT calculations. This is so, since to compute the transformed value for each position in the matrix, we need to account for all locations in the matrix to be processed. Hence, the original matrix is divided into a set of smaller 8×8 submatrices, each known as a block. These blocks are fed sequentially to the DCT, which transforms each block separately (Fig. 17.5). We then apply forward DCT.

If the input 2D matrix is $f(x,y)$ and the transformed matrix is $C(u,v)$, then for each of 8×8 blocks, we have from eqn. (17-6) where $N=8$

$$C(u,v)=\frac{1}{4}\alpha(u)\,\alpha(v)\sum_{x=0}^{7}\sum_{y=0}^{7}f(x,y)\cos\left[\frac{(2x+1)u\pi}{16}\right]\cos\left[\frac{(2y+1)v\pi}{16}\right] \qquad (17-8)$$

$$\alpha(u),\alpha(v)=\begin{cases}\frac{1}{\sqrt{2}} & u,v=0\\ 1 & u,v \quad otherwise\end{cases} \qquad (17-9)$$

where x,y,u,v all vary from $0 to 7$.

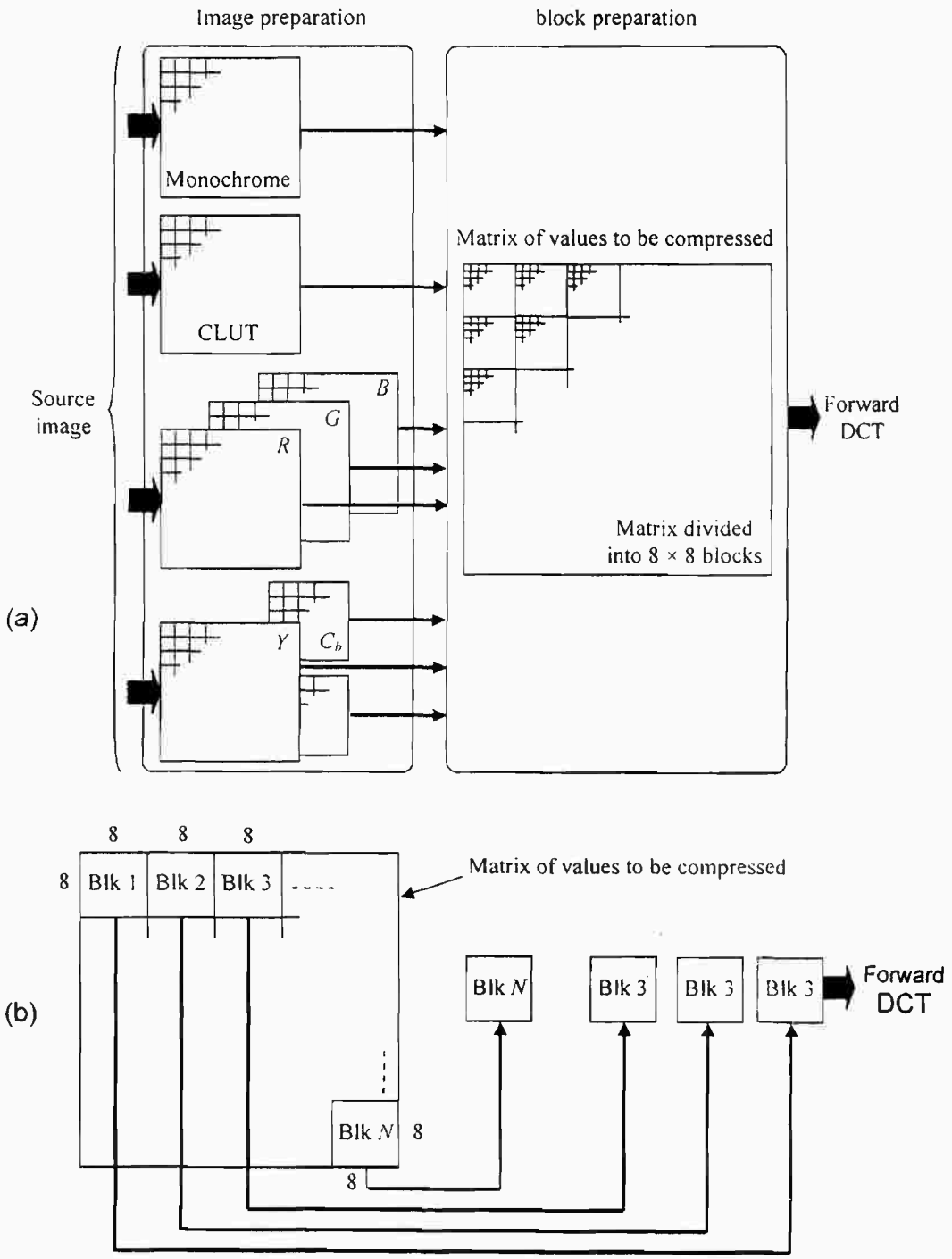


Fig. (17.5) Image/Block preparation
a) image preparation b) block preparation

We note that all 64 values of the input matrix contribute to each entry in the transformed matrix$C(u,v)$. For $u=v=0$, the value in location $C(0,0)$ of the transformed matrix is simply a function of the summation of all values in the input matrix, i.e., the sum of all 64 values in the matrix, hence, is called the DC coefficient. Since the values in all other locations of the transformed matrix have frequency components, they are called AC coefficients. For $y=0$ and $x=1-7$, we have horizontal frequencies. For $x=0$ and $y=1-7$, we have vertical frequencies. For $x=1-7$ and $y=1-7$, we have both. For $v=0$, only horizontal frequency coefficients are present, which increase in v frequency for$u=1-7$. For $u=0$, only vertical frequency coefficients are present which increase in frequency for$v=1-7$. In all other locations in $C(u,v)$ matrix, both horizontal and vertical frequency coefficients are present.

Considering a typical 640×480 pixels image and block size 8×8, we have 4800 blocks. In general, neighboring blocks will have DC coefficients more or less the same, and only a few AC coefficients. In regions of the picture, where there is sharp variations, the DC coefficients of the blocks will vary and the number of appreciable AC coefficients will increase (Fig. 17.6).

Since the human eye responds primarily to DC coefficient and the lower spatial frequency components, the magnitude of some of the frequency components may be below a certain threshold, hence the eye will not detect them. So we may drop them, i.e., set their magnitudes to O without much loss in information. Those frequency components will be irreversible by the decoder. This procedure will curtail the bandwidth without significant cost.

In addition to determining whether a particular frequency coefficient is above a defined threshold, we may use the thresholding procedure to cut down on the information content by setting a variable threshold value for the DC coefficient and AC coefficients so that what has to be sent is a comparison between the value at each location of the transformed matrix and the corresponding threshold at that location. This process is called quantization. Thus we reduce the size of the large DC and AC coefficients that have to be transmitted and drop those AC coefficients that may be neglected. Moreover, instead of simply comparing each coefficient with the corresponding threshold value, a division operation is carried out, using the defined threshold value at the corresponding location in the transformed matrix as or divisor. If the quotient can be rounded to zero, then the coefficient is less than the threshold. If it is nonzero, then it indicates the number of times this coefficient is larger than the threshold.

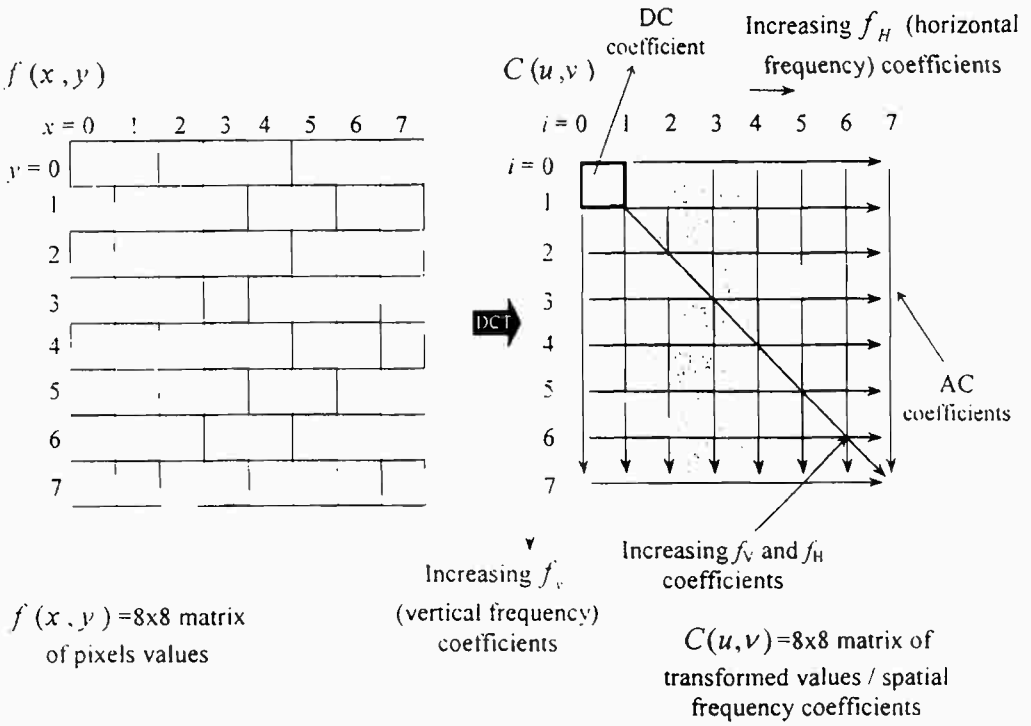


Fig. (17.6) DC and AC coefficients

By sending such information rather than the absolute values of the coefficients we further cut back on the bandwidth. If the division is, e.g. 16 this procedure saves already 4 bits over the use of the absolute value. Of course this saving is at the expense of precision. As long as this can be tolerated, it is a welcome step. Of course, the decoder multiplies the received values by the corresponding threshold value.

Since the sensitivity of the eye varies with spatial frequency, the threshold values vary for each of the 64 DCT coefficients. Those values are held in a 2D matrix called quantization table indicating the threshold value to be used for the DCT coefficient in the corresponding position in the matrix. The choice of such values is a compromise between precision and saving on bandwidth. This table must be known at both the encoder and decoder. Default tables exist for Y and C_r,C_b, but customized tables can also be designed but must be made available to the decoder.

Fig. (17.7) shows a set of threshold values, a given quantization table and the resulting quantized values (prob. 17.3). We should note that rounding is performed to the nearest integer. The threshold values increase in magnitude with spatial frequency. The DC coefficient is the largest. Many of the higher

frequency coefficients in the digitized coefficient matrix are zero. These properties are the mainstay for compression in JPEG, which is made use of the following entropy encoder phase. In the entropy encoder, four steps are involved:

1. Vectoring.
2. Differential encoding.
3. Runlength encoding.
4. Huffman encoding.

Entropy encoding operates on a 1D string of values, i.e., vector. The output of the quantization stage is a 2D matrix of values. Hence, before we apply entropy encoding, we must first represent the values as a 1D vector. If we scan the 2D matrix line by line we have 1×64 vectors, consisting of an irregular mix of nonzero and zero values. In order to make use of the presence of a large number of zero in the quantized matrix, zigzag scan of the quantized matrix is used. The DC coefficient and lower frequency AC coefficients in both horizontal and vertical directions are scanned first. All the higher frequency coefficients come in sequential order, making the next step of compression easier (Fig. 17.8).

The DC coefficient is the average value of the image and is the largest. Since from block to block it is not expected that the DC coefficient varies dramatically, it is worth it not to send the DC coefficient for every block, but rather send the first DC coefficient and subsequently scan only the differences (positive or negative). In this way, further cut down on the number of bits is achieved. If for example the sequence of DC coefficients in consecutive quantized blocks are 12,13,11,11,10, then the corresponding difference values are 12,1,-2,0,-1, noting that the differences are measured from the immediate predecessor and the first difference value is relative to zero. The difference values are then encoded in the form (SSS, value), where SSS field is the number of bits needed to encode the value, and the value field is the actual bits that represent the value. The rules to encode each value is given in Table (17.1). We see that the number of bits required to encode each value is determined by its magnitude. A positive value is encoded using the unsigned binary form and a negative value by the complement of this. The value of zero is encoded using a single 0 bit in the SSS field (prob. 17.5).

Now the remaining 63 values in the vector are AC coefficients. Because of the zigzag scan, the vector contains long strings of zeros. AC coefficients are encoded in the form of a string of pairs of values. Each pair is made up of (skip, value). The skip field is the number of zeros in the run and the value is the next nonzero coefficient. When the final pair is (0,0), the remaining coefficients in the block are zeros (Prob. 17.6).

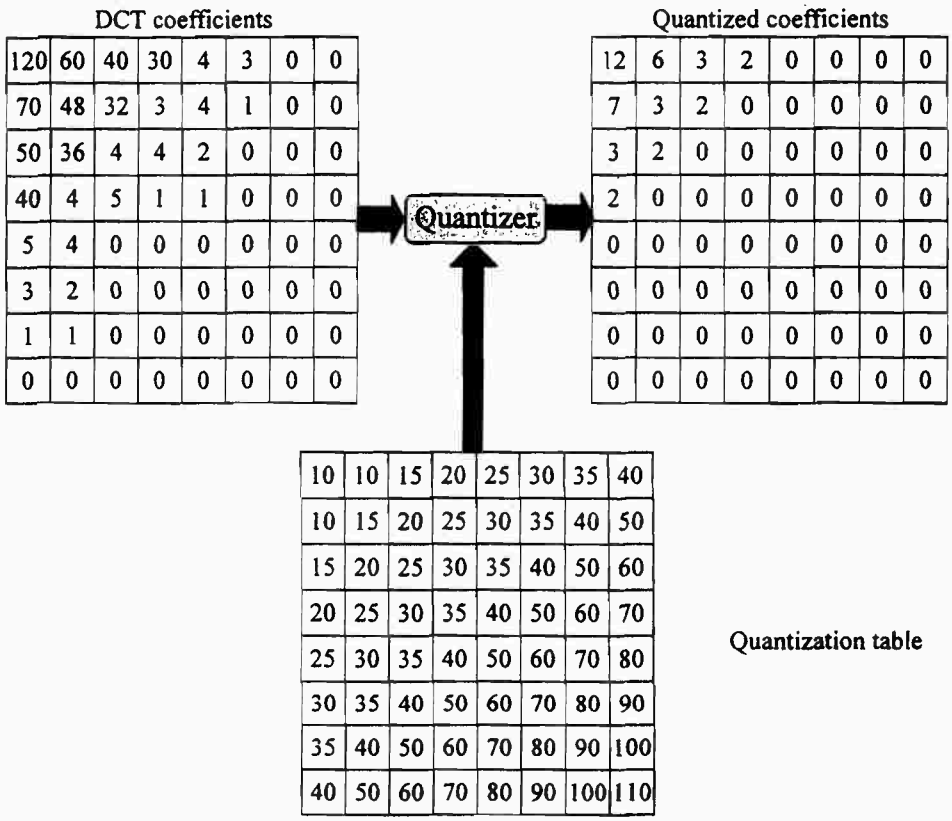


Fig. (17.7) Quantized DCT coefficients

Table (17.1) JPEG encoding

Difference value	Number of bits needed (SSS)	Encoded value	
0	0		
−1, 1	1	1 = 1	−1 = 0
−3, −2, 2, 3	2	2 = 10	−2 = 01
		3 = 11	−3 = 00
−7.. −4, 4.. 7	3	4 = 100	−4 = 011
		5 = 101	−5 = 010
		6 = 110	−6 = 001
		7 = 111	−7 = 000
−15.. −8, 8.. 15	4	8 = 1000	−8 = 0111
		⋮	

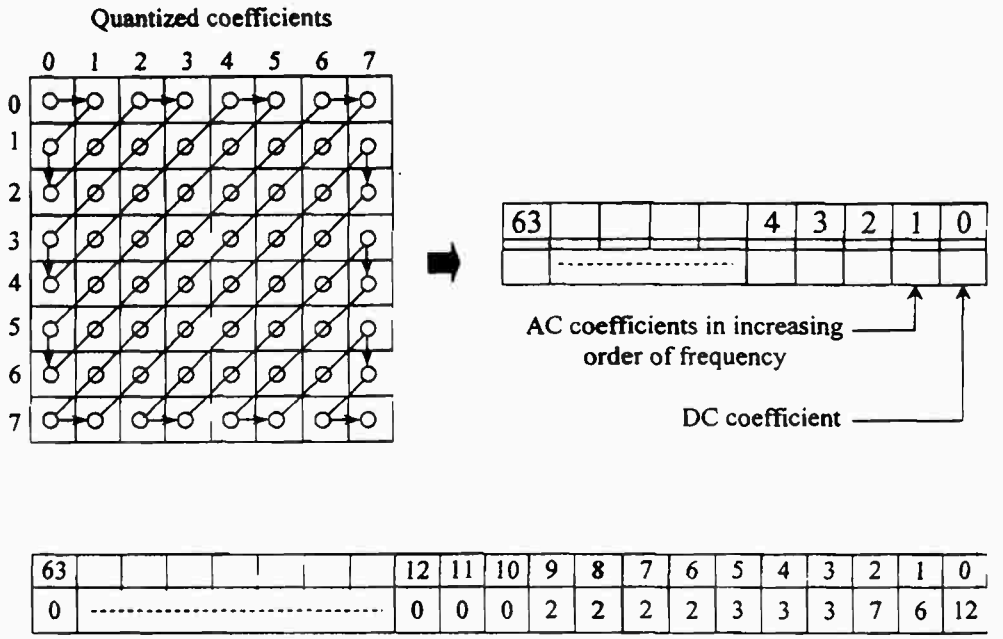


Fig. (17.8) Zigzag scan

Table (17.2) Default Huffman codewords

Number of bits needed (SSS)	Huffman codeword
0	010
1	011
2	100
3	00
4	101
5	110
6	1110
7	11110
11	111111110

The value field is encoded in the form SSS / value (Prob. 17.7). Instead of sending long strings of binary digit, much shorter codewords are used. The length of each codeword is a function of its relative frequency of occurrence. A table of codewords is used (Table 17.2) as an example. This procedure is called variable length coding.

The same method is used for both differential and runlength encoder outputs. For the differential encoded DC coefficients, the bits in the SSS field are sent in Huffman encoded form.

The prefix property inherent in Huffman code enables the decoder to determine unambiguously the first SSS field from the received encoded bit stream. For each of the runlength encoded AC coefficients in the block the bits that make up the skip and SSS fields are treated as a component symbol and is then encoded in the generalized default Huffman code (Table 17.3). The Huffman encoded bit stream is obtained by adding the runlength value to each of the Huffman codewords.

To decode the received bit stream, the receiver first searches the bit stream starting at the leftmost bit for a valid codeword, and then it determines the skip and the SSS fields from the Huffman table. The SSS field is then used to determine the number of bits in the runlength value field and so on until (0,0) codeword (called end of bit stream EOB) is received.

Table (17.3) Generalized default Huffman codewords

Skip/SSS	Codeword	Skip/SSS	Codeword	Skip/SSS	Codeword
0/0	1010 [= *EOB*]	3/3	1111111011 1	6/6	1111111110 101010
0/1	00	3/4	1111111110 010000	6/7	1111111110 101011
0/2	01	3/5	1111111110 010001	6/8	1111111110 101100
0/3	100	3/6	1111111110 010010	6/9	1111111110 101101
0/4	1011	3/7	1111111110 010011	6/10	1111111110 101110
0/5	11010	3/8	1111111110 010100	7/1	11111001
0/6	111000	3/9	1111111110 010101	7/2	1111111100 1
0/7	1111000	3/10	1111111110 010110	7/3	1111111110 101111
0/8	1111110110	4/1	111011	7/4	1111111110 110000
0/9	1111111110 000010	4/2	1111111000	7/5	1111111110 110001
0/10	1111111110 000011	4/3	1111111110 010111	7/6	1111111110 110010
1/1	1100	4/4	1111111110 011000	7/7	1111111110 110011
1/2	111001	4/5	1111111110 011001	7/8	1111111110 110100
1/3	1111001	4/6	1111111110 01101	7/9	1111111101 10101
1/4	111110110	4/7	1111111110 011011	7/10	1111111110 110110
1/5	1111111010	4/8	1111111110 01110	8/1	11111010
1/6	1111111100 00101	4/9	1111111110 011101	8/2	1111111110 00000
1/7	1111111110 000101	4/10	1111111110 011110	8/3	1111111101 110111
1/8	1111111110 000110	5/1	1111010	8/4	1111111110 111000
1/9	1111111110 000111	5/2	1111111001	8/5	1111111110 111001
1/10	1111111110 001000	5/3	1111111110 011111	8/6	1111111110 111010
2/1	11011	5/4	1111111110 100000	8/7	1111111110 111011
2/2	11111000	5/5	1111111110 100001	8/8	1111111110 111100

Skip/SSS	Codeword	Skip/SSS	Codeword	Skip/SSS	Codeword
2/3	1111110111	5/6	1111111110 100010	8/9	1111111110 111101
2/4	1111111110 001001	5/7	1111111110 100011	8/10	1111111101 1110
2/5	1111111110 001010	5/8	1111111110 100100	9/1	111111000
2/6	1111111110 001011	5/9	1111111110 100101	9/2	1111111110 111111
2/7	1111111110 001100	5/10	1111111110 100110	9/3	1111111111 000000
2/8	1111111110 001101	6/1	1111011	9/4	1111111111 000001
2/9	1111111110 001110	6/2	1111111100 0	9/5	1111111111 000010
2/10	1111111110 001111	6/3	1111111110 100111	9/6	1111111111 000011
3/1	111010	6/4	1111111110 101000	9/7	1111111111 000100
3/2	111110111	6/5	1111111110 101001	9/8	1111111111 000101
9/9	1111111111 000110	11/10	1111111111 011001	14/1	1111111101 10
9/10	1111111111 000111	12/1	111111010	14/2	1111111111 101100
10/1	111111001	12/2	1111111111 011010	14/3	1111111111 101101
10/2	1111111111 001000	12/3	1111111111 011011	14/4	1111111111 101110
10/3	1111111111 001001	12/4	1111111111 011100	14/5	1111111111 101111
10/4	1111111111 001011	12/5	1111111111 011101	14/6	1111111111 10000
10/5	1111111111 001011	12/6	1111111111 01110	14/7	1111111111 10001
10/6	1111111111 001100	12/7	1111111111 011111	14/8	1111111111 110010
10/7	1111111111 001101	12/8	1111111111 1000000	14/9	1111111111 110011
10/8	1111111111 001110	12/9	1111111111 100001	14/10	1111111111 10100
10/9	1111111111 001111	12/10	1111111111 100010	15/0	1111111101 11
10/10	1111111111 010000	13/1	1111111101 0	15/1	1111111111 110101
11/1	111111010	13/2	1111111111 100011	15/2	1111111111 110110
11/2	1111111111 010001	13/3	1111111111 100100	15/3	1111111111 110111
11/3	1111111111 010010	13/4	1111111111 100101	15/4	1111111111 111000
11/4	1111111111 010011	13/5	1111111111 100110	15/5	1111111111 111001
11/5	1111111111 010100	13/6	1111111111 100111	15/6	1111111111 111010
11/6	1111111111 010101	13/7	1111111111 101000	15/7	1111111111 111011
11/7	1111111111 010110	13/8	1111111111 101001	15/8	1111111111 111100
11/8	1111111111 010111	13/9	1111111111 101010	15/9	1111111111 111101
11/9	1111111111 011000	13/10	1111111111 101011	15/10	1111111111 1111110

The bit stream output of a JPEG encoder is stored in the memory of a computer, ready to be released. The JPEG standard must include a definition of the structure of the total bit stream relating to an image. This is known as frame builder. The frame builder (Fig. 17.9) encapsulates all the information relating to a particular image. At the top level, the complete frame plus header is encapsulated between the start of frame and the end of the frame to allow the receiver to determine the start and end of all information relating to a complete image. The frame header contains the following fields.

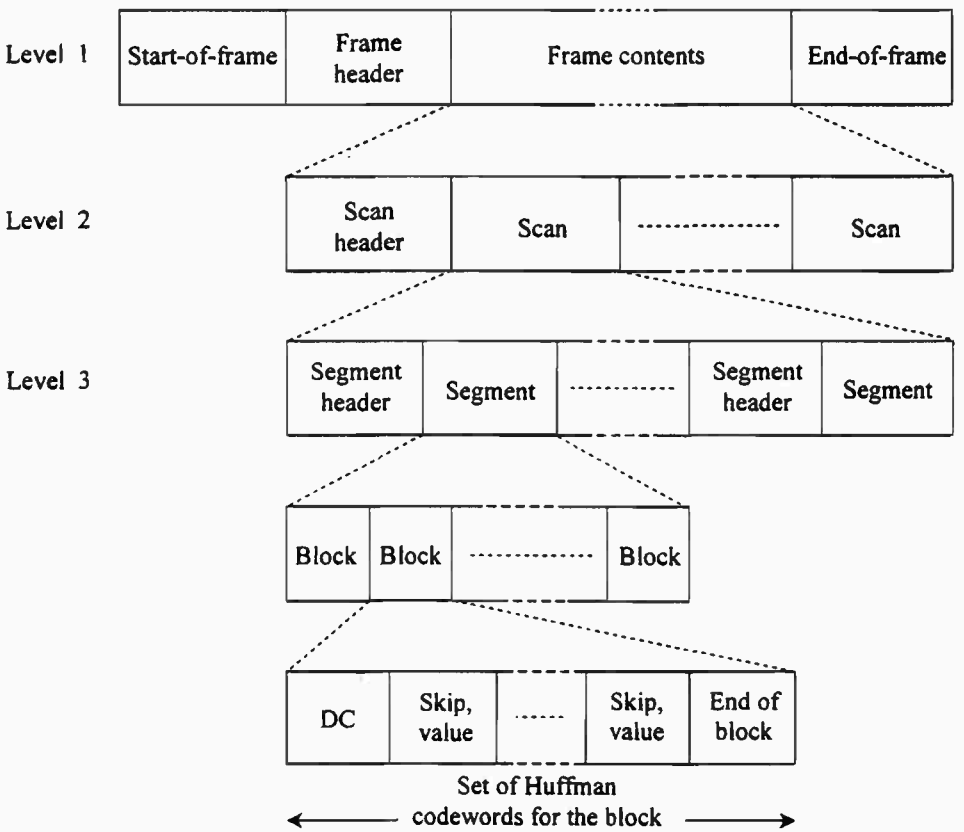


Fig. (17.9) JPEG encoder output bit stream format

1. The overall width and height of the image in pixels.
2. The number and type of components that are used to represent the image (CLUT, R/G/B, $Y/C_r/C_b$).
3. The digitization format used (4:2:2,4:2:0)

At the second level, frames consist of a number of components, each of which is known as a scan. These are also preceded by a header containing fields that include:

1. Identity of the components.
2. Number of bits used to digitize each component.
3. The quantization table needed to encode each component.

Each scan consists of segments, each of which contains a group of 8×8 blocks preceded by a header. This contains the Huffman table of values that have been used to encode each block in the segment in case the generalized default table of values were not used. Thus, each segment may be decoded independently of the others. This eliminates propagation of error. The JPEG decoder (Fig. 17.10) performs the reciprocal task of the JPEG encoder (Fig. 17.4).

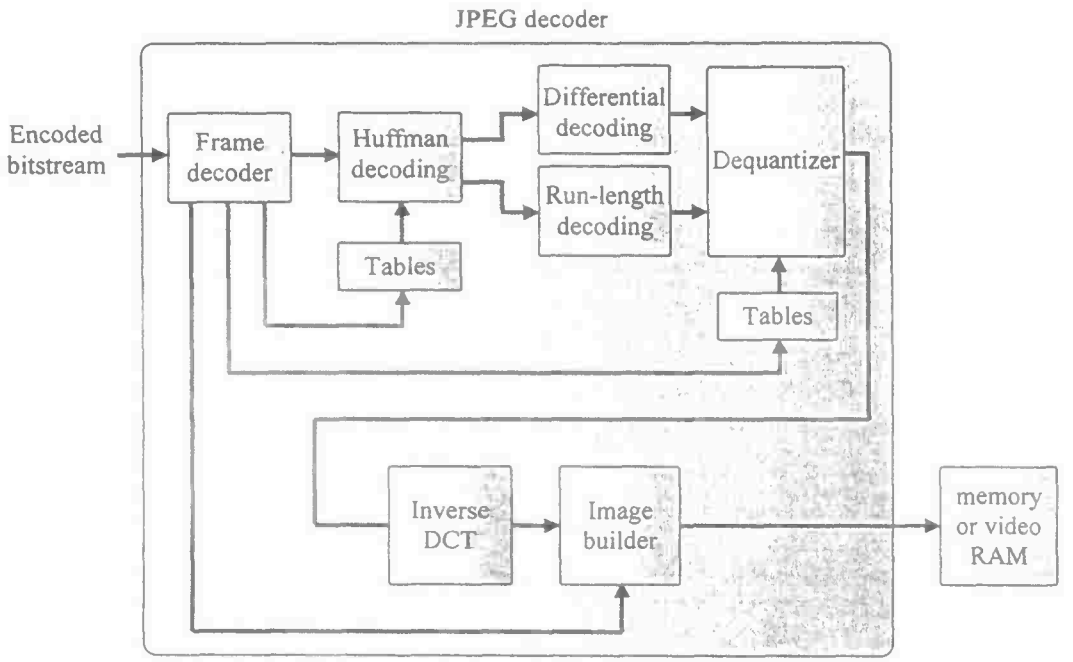


Fig. (17.10) JPEG decoder

As the bit stream is received, the frame decoder identifies the control information and tables within the various headers and pass the control information to the image builder. It then passes the compressed bit stream into the Huffman decoder for decompression, using either the default table or the one preloaded. The two decompressed streams containing the DC and AC coefficients of each block are then passed to the differential and runlength decoder. The resulting matrix of values is then dequantized. Each resulting block of 8×8 spatial frequency coefficients is then passed into an inverse DCT to obtain the expression.

$$f(x, y) = \frac{1}{4} \sum_{u=0}^{7} \sum_{v=0}^{7} \alpha(u)\, \alpha(v) C(u, v) \cos\left[\frac{(2x+1)u\pi}{16}\right] \cos\left[\frac{(2y+1)v\pi}{16}\right] \quad (17-10)$$

$$\alpha(u), \alpha(v) \begin{cases} = \frac{1}{\sqrt{2}} & u, v = 0 \\ = 1 & u, v \quad otherwise \end{cases} \quad (17-11)$$

The image builder then reconstructs the original image from these blocks, using the control information passed to it by the frame decoder.

JPEG can achieve a compression of 10 – 20 or even more (particularly if CLUT is used). Not only that JPEG saves on the memory size and the bandwidth, but it reduces the time delay incurred while accessing an image. It is also possible to encode and rebuild the image in a progressive way, by first

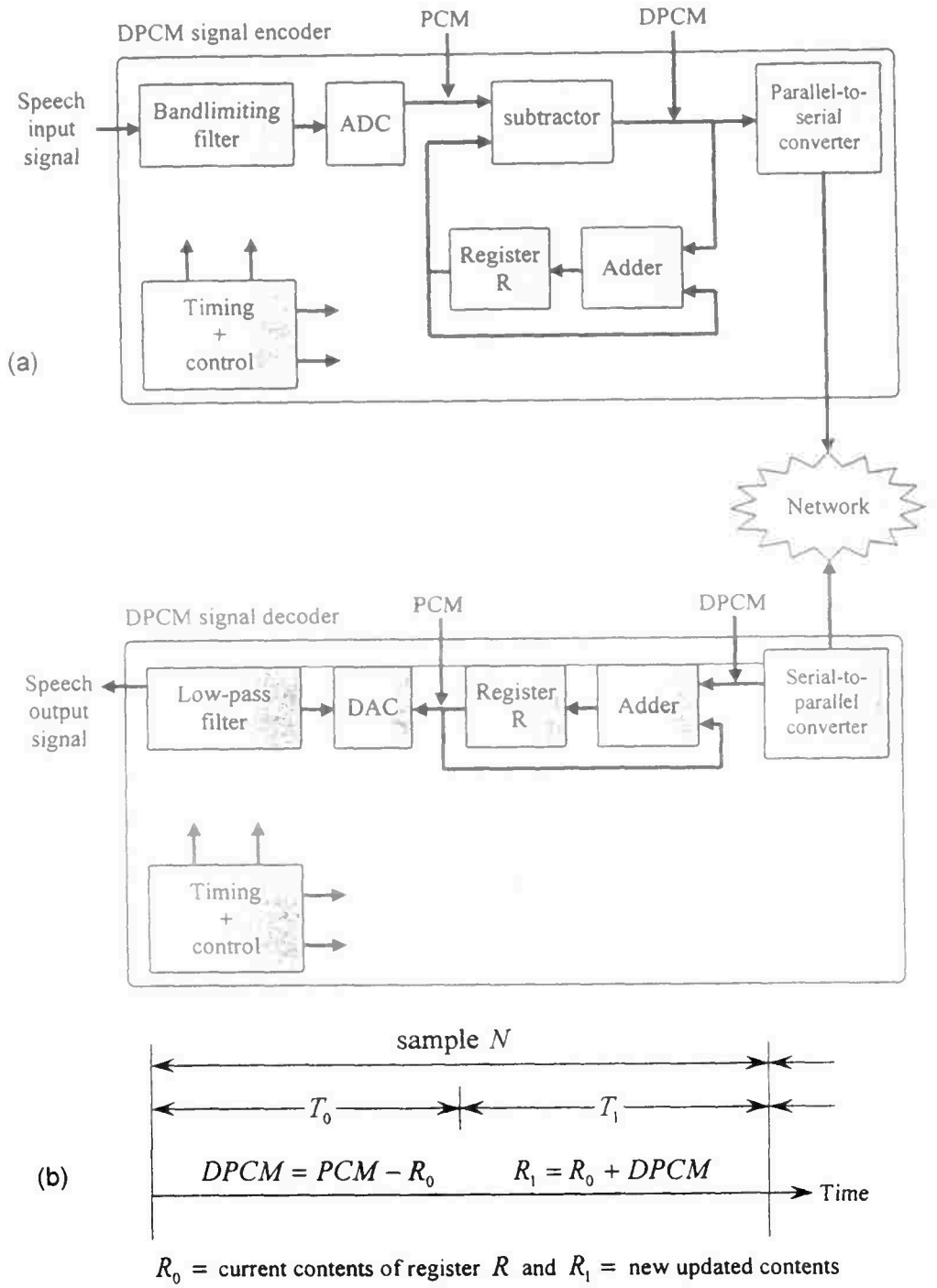
sending an outline of the image and then progressively adding more details to it. In the progressive mode first the DC and low frequency coefficients of each block are sent, then the higher frequency coefficients. Alternately (hierarchical mode) the total image is first sent at a low resolution (320×240), then at a higher resolution (640x480).

17.4 Audio Compression:

Audio signals are sampled and digitized by PCM. The sampling rate is twice the maximum frequency component. In the case when the bandwidth of the channel is less than the maximum frequency component, then the sampling rate is twice the bandwidth of the channel (band-limited signal). For speech, the maximum frequency component is $10kHz$ and the sampling rate is $20kHz$, and for music the maximum frequency component is $20kHz$ and the sampling rate is $40kHz$. The number of bits is 12 bits for speech and 16 bits for general audio. Thus, for speech, the bit rate is 240 kb/s, and for stereophonic music $1.28Mbps$. In practice, however, the channel bandwidth dictates values less than these. This can be done either by sampling at a lower rate (or using fewer bits per sample) or by a compression algorithm. The first approach results in the degradation of the quality of the retrieved signal in the decoder due to the loss of high frequency components altogether. Also, the use of fewer bits per sample increases the quantization noise. Thus, a compression algorithm is used which achieves a perceptual quality comparable to that obtained by high rate sampling and consistent with the ear sensitivity, but at a reduced bandwidth. The first step in compression is the use of differential pulse code modulation (DPCM) instead of PCM. Thus, instead of sending the bits per sample of every time, we send only the difference from one sample to the next (Fig. 17.11).

The previous digitized sample is held in register R, and the difference signal is computed by subtracting the current contents of the register R_o from the new digitized sample output by the ADC. The value in the register is then updated by adding to the current register contents the computed difference signal output by the subtractor. The decoder adds the received difference signal to the previously computed signal held in the register.

Additional savings in bandwidth or improved quality can be obtained by varying the number of bits used for the difference signal, depending on its amplitude, i.e., use small number of bits for small amplitudes. This is called adaptive differential PCM (ADPCM). Another technique is called subband coding (Fig. 17.12). This can extend the $3.4kHz$ limit for input speech to $7kHz$. The audio input signal is first passed through two filters, one for the range $50-3.5kHz$, and the other for the range $3.5kHz-7kHz$.



R_0 = current contents of register R and R_1 = new updated contents

Fig. (17.11) DPCM

a) encoder / decoder b) encoder timing

Thus, the input signal is effectively divided into two separate equal - bandwidth segments; the lower subband and the upper subband. Each is then sampled and encoded independently using ADPCM. The sampling rate of the upper subband being 16 k sample/s to allow for the presence of the higher frequency components.

The use of two subbands makes it possible to use different bit rate for each. The ear is more sensitive to the lower frequency components. The lower subband may use a bit rate of $48kbps$, while the upper subband uses $16kbps$. Thus, for $64kbps$, we favor the lower subband. The multiplexer in the encoder and the demultiplexer in the decoder are set such that the receiver can divide them again into two separate streams for decoding.

A new approach stimulated by the advent of low cost digital signal processing chips focuses on analyzing the audio waveform to determine a selection of the perceptual features it contains. These features are then quantized and sent, instead of quantizing and sending the input audio. The decoder uses these features together with a sound synthesizer to regenerate a sound audio signal. This is called linear predictive coding (LPC). With this, very high levels of compression, and hence low bit rates can be achieved. The features to be determined are:

a) Pitch which is the frequency perception in the ear. The ear is more sensitive in the range 2-5 kHz.
b) Period which is the duration of the signal.
c) Loudness which is the amplitude of signal or the level of the energy of the signal.

Additionally, there are vocal tract excitation parameters which are:

1. Voiced sound: sounds generated through the vocal chords (letters m, v, l).
2. Unvoiced sounds: sounds generated when the vocal chords are open (letters f, s).

The source waveform is analyzed to determine these parameters. It is then possible to use there parameters together with a suitable model of the vocal tract to generate a synthesized version of the original speech signal.

In the LPC encoder (Fig. 17.13a), the input speech waveform is first sampled and quantized at a defined rate. A block of digitized samples (segment) is then analyzed to determine the various perceptual parameters.

The speech signal generated by the vocal tract model in the decoder (Fig. 17.13b) is a function of the present output of the speech synthesizer according to the current set of model coefficients, in addition to a linear combination of the previous set of model coefficients. Thus, the vocal tract model is adaptive. The encoder sends a new set of coefficients for each segment. The encoder must also send fields for pitch, loudness, period, voiced and unvoiced information.

A more sophisticated version of LPC is called code excited linear prediction (CELP). In the CELP model, instead of treating each digitized segment independently for encoding purposes, only a limited set of segments is used, each called waveform template.

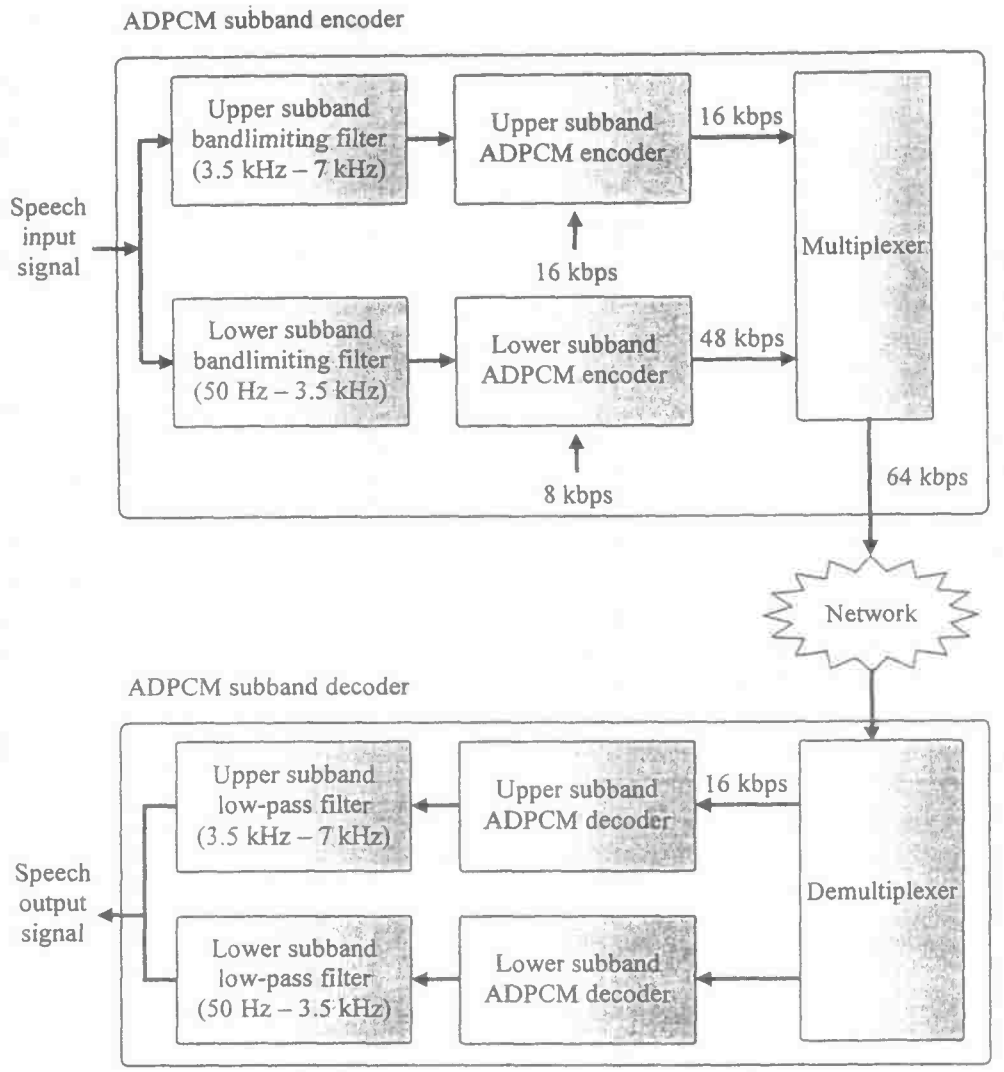


Fig. (17.12) Subband coding
a) encoder b) decoder

A precomputed set of templates is made available to both the encoder and decoder in a template codebook. Each of the individual digitized samples that make up a particular template in the codebook are differentially encoded. Each codeword sent selects a particular template from the codebook whose difference values best match those quantized by the encoder. In this way, there is continuity from one set of samples to another, and hence, an improvement in sound quality.

Both LPC and CELP are used primarily for telephony applications, and hence, the compression of a speech signal.

Another type of compression for general audio, such as with digital TV broadcast is called perceptual encoding. A psycho acoustic model is used which exploits a number of limitations of the human ear. Sampled segments are analyzed, and only those features perceptible to the ear are transmitted. Although the human ear is sensitive to frequencies $15Hz - 20kHz$, yet, the sensitivity of the ear is nonlinear. The ear is more sensitive to some frequencies than to others. Also, in the presence of multiple signals, the strongest signal masks other signals, i.e., lowers the level of sensitivity (making the ear less sensitive) to weak signals. This is called frequency masking. Also, when the ear receives a loud sound, it takes a while before the ear can hear a quiet sound. This is called temporal masking. The psycho-acoustic model identifies those signals that are affected by others and eliminates from transmission signals that the ear is not sensitive to. This reduces the amount of information to be transmitted. We may define the dynamic range of the ear as the ratio of the loudest sound it can hear to the quietest sound and is about $96dB$.

In the presence of a single frequency, the perception threshold of the ear, i.e., the minimum level of sensitivity as a function of frequency is shown (Fig. 17.14a). The ear is most sensitive in the range 2-5 kHz . If two signals A,B are present, A above threshold would be heard, while B would not. Now if signal B is made stronger while signal A is still present, the strengthening of signal B causes a perturbation in the threshold curve such that signal A is now below the modified threshold and would not be heard. This deformation of the sensitivity curve by a strong signal is the frequency masking effect (Fig. 17.14b). This deformation also varies depending on the masking frequency (Fig. 17.15). It is seen that the width of the masking curve in the range of frequencies that are affected (called critical bandwidth) varies with frequency. (Fig. 17.16) shows the effect of temporal masking. After the loud sound ceases, it takes a short period of time (tens of milliseconds) for the signal amplitude to decay. During this time, signals whose amplitude is below the decay envelope will not be heard, and hence, need not be transmitted.

17.5 MPEG Audio Coder:

The time varying audio input signal is first sampled and quantized using PCM. The bandwidth that is available is divided into a number of subbands using a bank of analysis filters. The filter bank maps each set of 32 PCM samples into an equivalent set of 32 frequency values (samples), one per subband, hence, called subband sample. For 32 PCM samples, we have 32 subbands, a sampling rate of 32 $kbps$, maximum signal frequency $16kHz$, each subband has a bandwidth of $500Hz$. Each 12 successive sets of 32 PCM or subband samples comprise a segment of $12 \times 32 = 384$ samples.

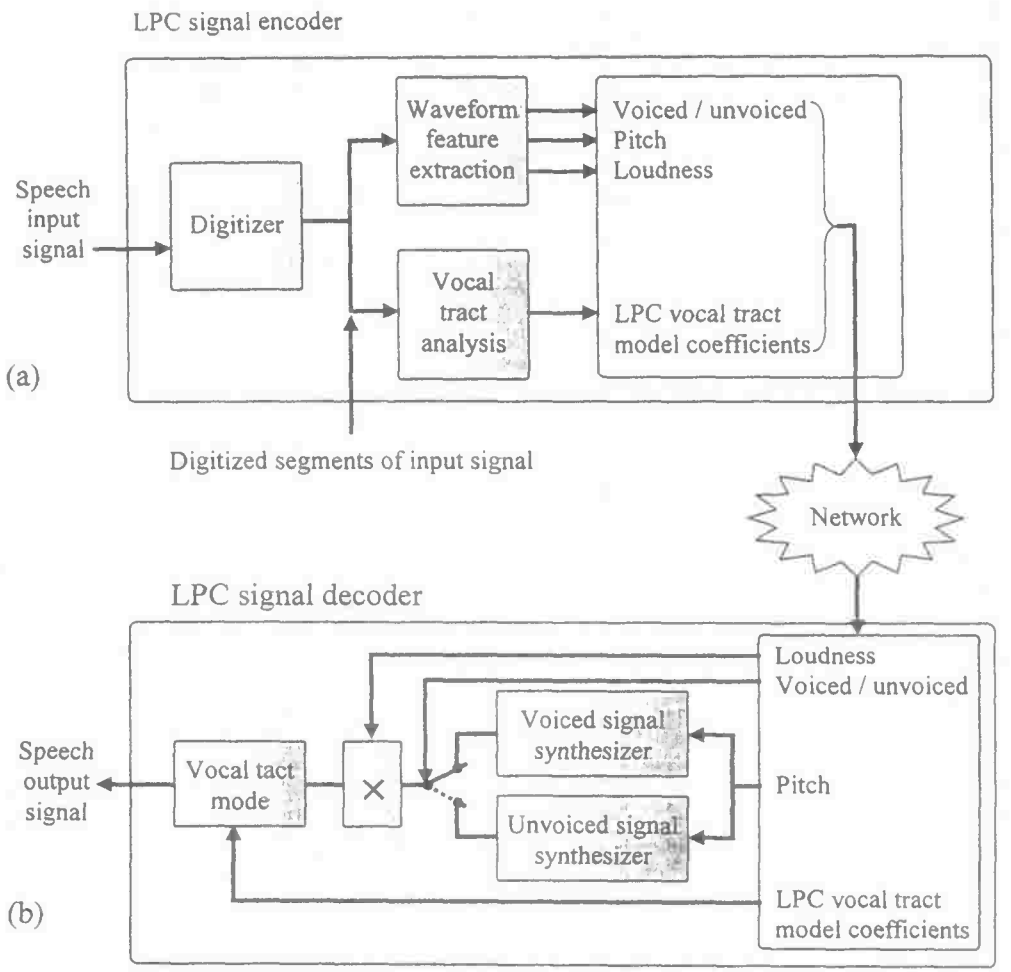


Fig. (17.13) Linear predictive coding LPC
a) encoder b) decoder

The filter bank determines also the maximum amplitude in corresponding subband in each segment. Mathematically, the 12 sets of 32 PCM samples are converted to an equivalent set of frequency components, using discrete Fourier transform (DFT) defined as

$$f(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x)\, e^{-j 2\pi u x / N} \qquad (17-12)$$

for $k = 0, 1, 2.. N-1$ and

$$f(x) = \sum_{u=0}^{N-1} F(u)\, e^{j 2\pi u x / N} \qquad (17-13)$$

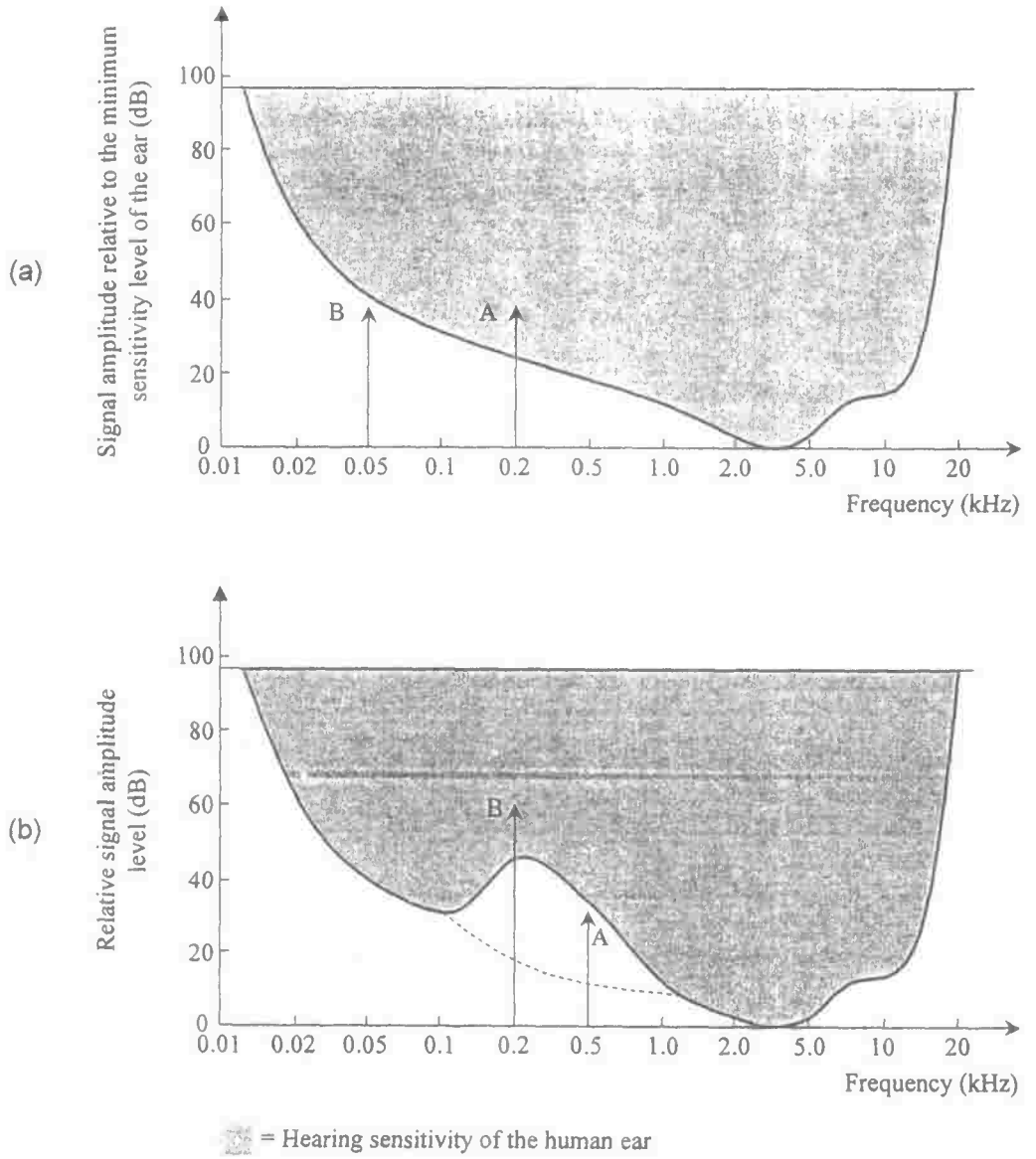


Fig. (17.14) Sensitivity of the ear

a) signals of equal strength. *b) frequency masking.*

This is based on discretizing $f(x)$ into a sequence $f(x_8)$, $f(x_0+\Delta x)$, $f(x_0+2\Delta x)$...

By taking N samples Δx apart so that

$$f(x)=f\left(x_0+x\,\Delta x\right) \qquad (17-14)$$

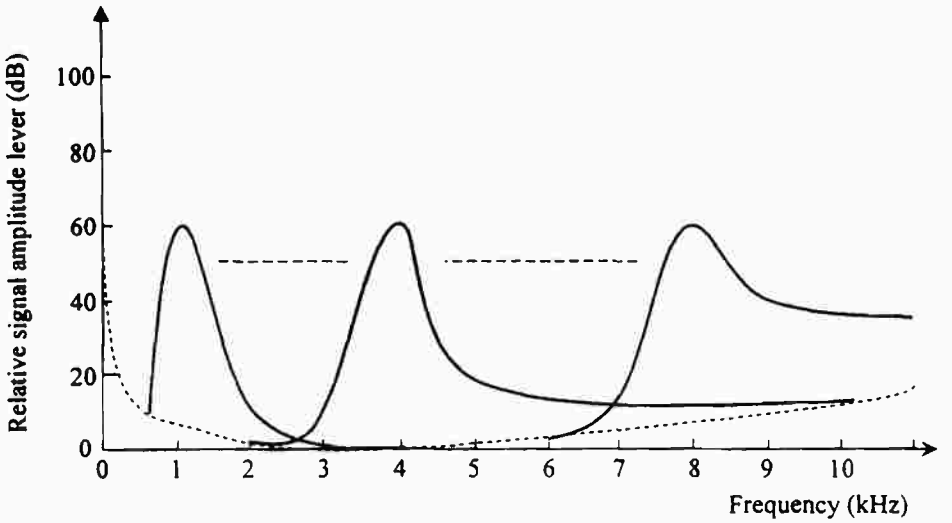


Fig. (17.15) Effect of frequency on masking effect

Thus $f(0)$, $f(1)$, . . . $f(N-1)$ denote N uniformly spaced samples from a corresponding continuous function. The values $u = 0,(1)2..N-1$ in the DFT correspond to samples at $0, \Delta u, 2\Delta u, ...(N-1)\Delta u$. The terms Δu and Δx are related by

$$\Delta u \ \Delta x = \frac{1}{N} \qquad (17-15)$$

Then, using the hearing thresholds and masking properties of each subband, the psychoacoustic model determines the various masking effects of the set. The output of the model is signal to mask ratio, which indicates those frequency components whose amplitude is below the audible threshold. Also, a set of scaling factors are used to determine the quantization accuracy, and hence the number of bits to be used for each of the audible components.

The frequency components in regions of highest sensitivity need quantization with more accuracy, i.e., more bits and less quantization noise. Fig. (17.17) shows the MPEG perceptual coder and frame format. The header contains information on the sampling frequency. The subband sampling SBS. involves nonlinear quantization.

In the decoder, the synthesis filter bank produces the corresponding set of PCM samples which are decoded to produce the time varying analog output segment. MPEG is used for digital audio cassette (32-448 $kbps$) of compressed audio digital audio and video broadcasting (32-192 $kbps$) of compressed audio and CD quality audio over low bit rate channels ($64kbps$) of compressed audio.

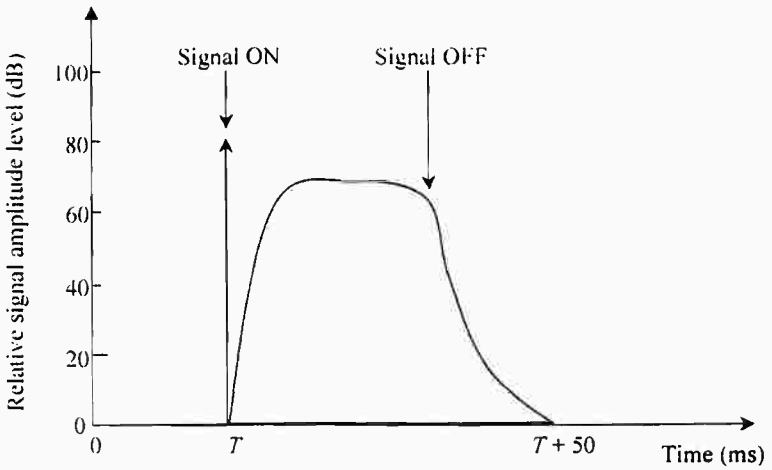


Fig. (17.16) Temporal masking

The psychoacoustic model in MPEG coder controls the quantization accuracy of each subband by computing and allocating the number of bits per samples. This bit allocation varies from one sample to another, and this information must be sent with the actual quantized samples, and is used by the decoder in the dequantization. This process is called forward adaptive bit allocation FABA (Fig. 17.18a).

Alternatively, a fixed bit allocation may be used for the encoder and decoder. Thus, the bit allocation may need not be sent along. This is the idea of basic Dolby (acoustic) coder (Fig. 17.18b). We could also allow adaptive bit allocation without having to transmit this information, if we let the decoder have the same psychoacoustic model as that in the encoder, or use a hybird model as in the case of digital broadcast.

17.6 Video Compression:

In principle, we may use JPEG algorithm to each frame of the video source. This is called moving JPEG (MJPEG). But the compression obtained by this method is not large enough to produce the compression ratios needed.

In practice, there is considerable redundancy not only spatially within the frame but also temporally from frame to frame, since a small portion of each frame is involved with motion. A typical scene in a movie has a minimum duration of 3 seconds, then at 60 frame/s, each scene is composed of 180 frames. Hence, by sending only information relating to those segments of each frame that have movement, thus, considerable saving in bandwidth can be achieved by taking only temporal differences into account.

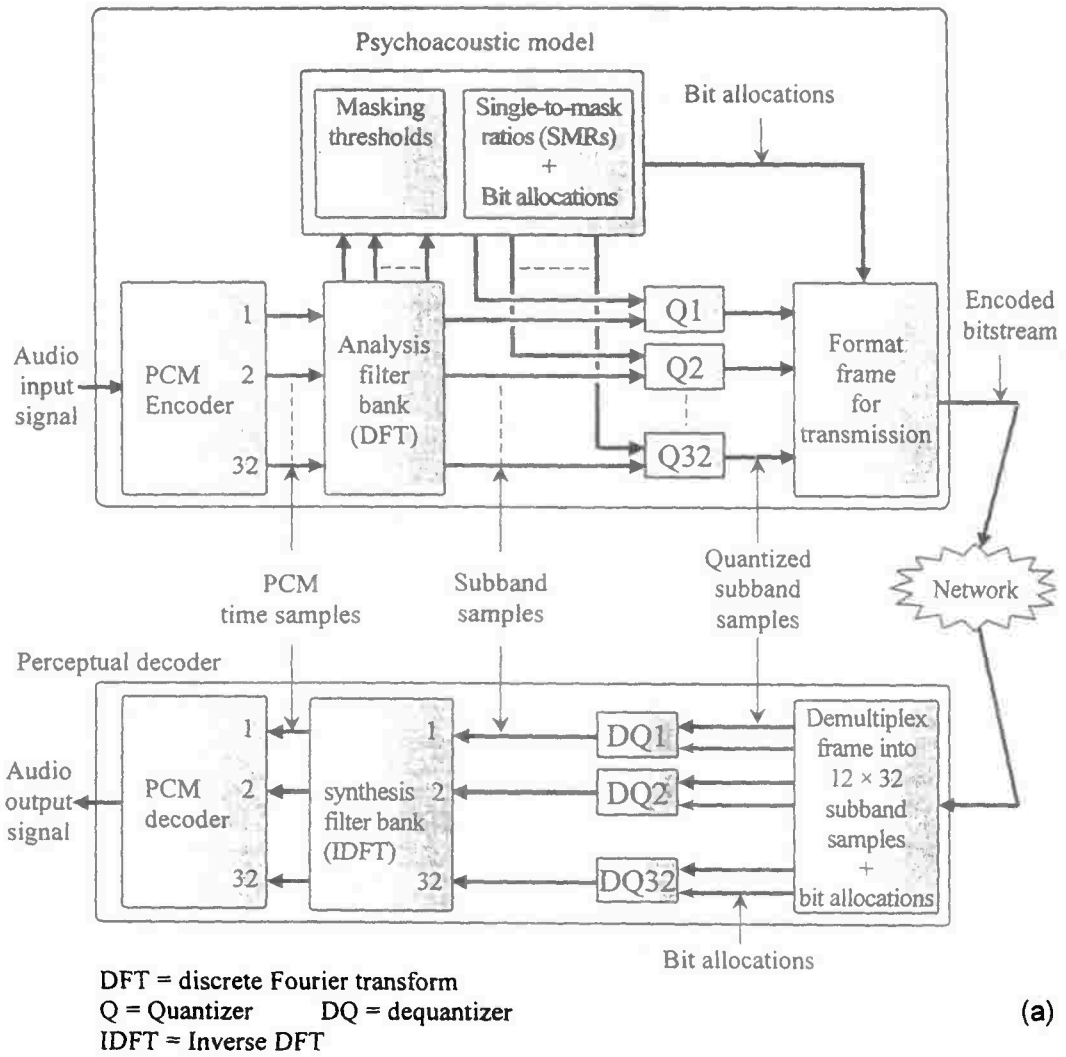


Fig. (17.17) MPEG perceptual coder
a) encoder / decoder b) frame format

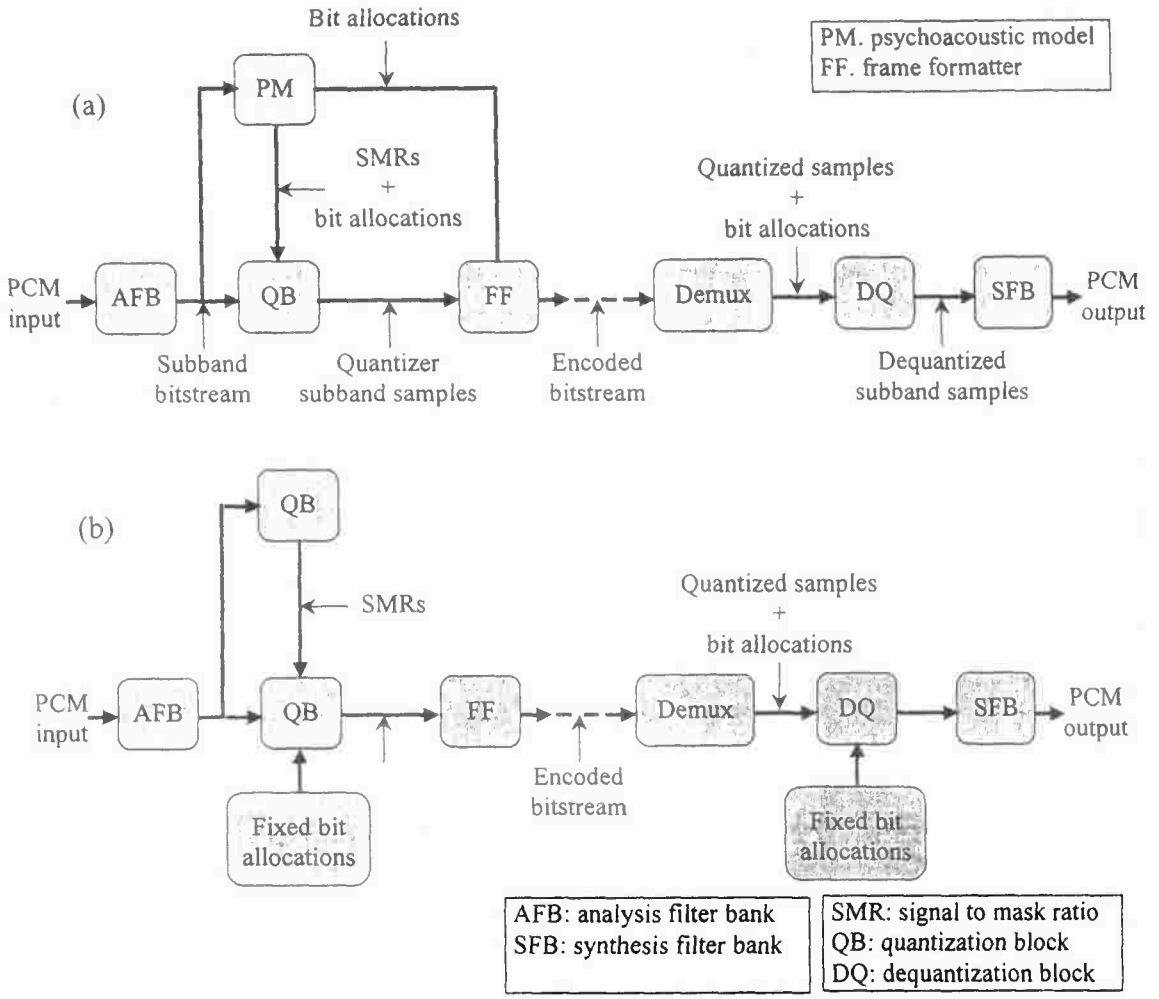


Fig. (17.18) Perceptual coder / Basic Dolby System
a) forward adaptive bit allocation b) fixed bit allocation

Instead of sending the source video as a set of individually compressed frames, just a selection is sent in this form, while for the remaining frames only the differences between the actual frame contents and the predicted frame contents are sent. The accuracy of the prediction operation is determined by how well any movement between successive frames is estimated. This is called motion estimation. For fine tuning of motion, additional information must be given to indicate any small differences between the predicted and actual positions of the moving segments. This is called motion compensation.

There are two basic types of compressed frames, those that are encoded independently (introduced or **I** frames), and those that are predicted (***P*** frames). There are two types of predicted frames, predictive ***P*** frames and bidirectional or ***B*** frames (intercoded or interpolated frames) (Fig. 17.19).

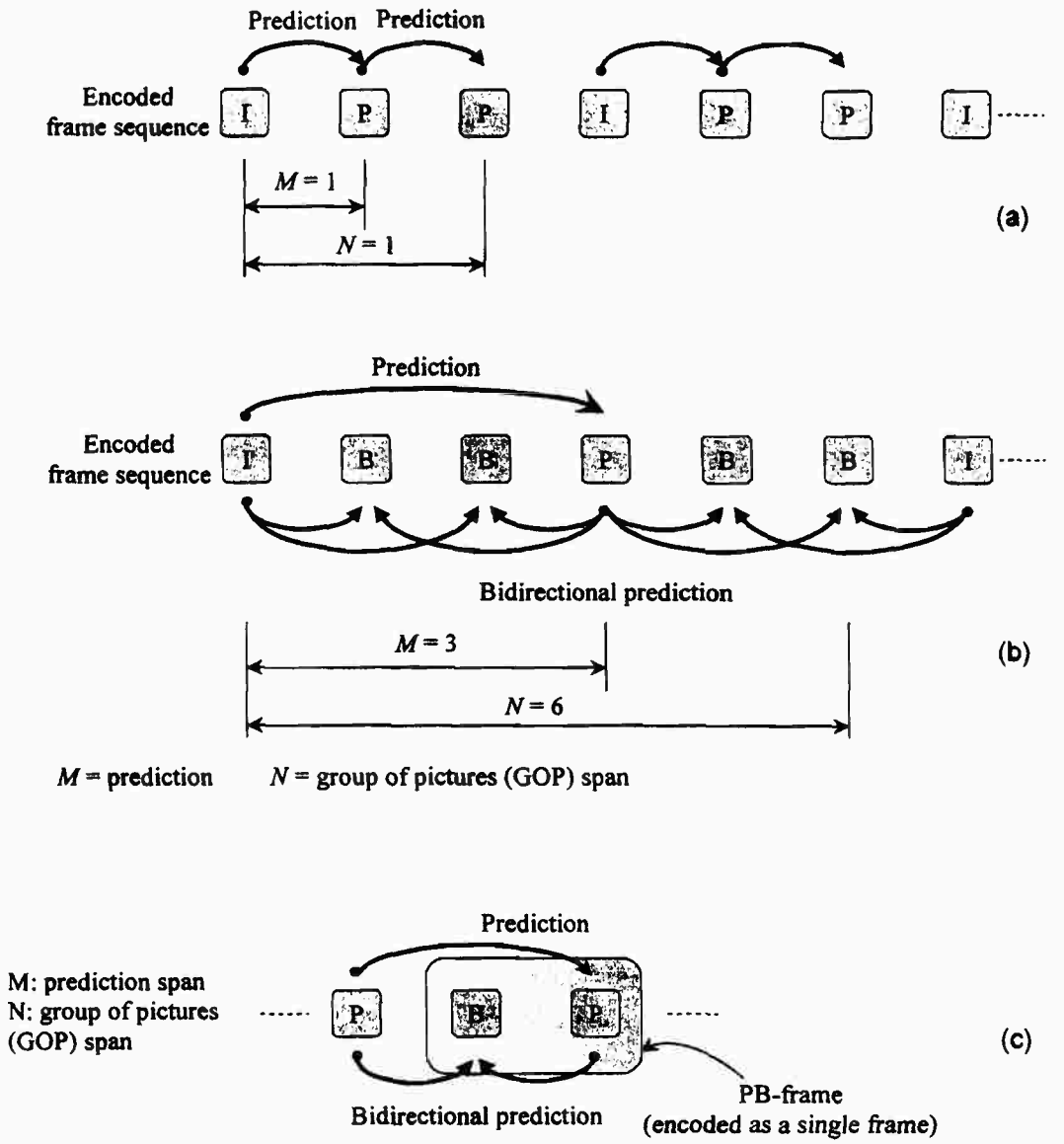


Fig. (17.19) Frame types

a) I and P frames b) I , P and B frames c) PB frames

The I frames (first frames of new scenes) are encoded without reference to any other frame. Each frame is treated as a Y, C_b and C_r matrices are encoded independently using JPEG algorithm.

The I frames must be sent at regular intervals to make up for possible corruption. Such corruption would affect all predicted frames which might cause a scene to be lost. The number of frames between successive I frames is called group of pictures (GOP) $N = 3-12$. The encoding of a P frame is relative to the contents of either a preceding I frame or a preceding P frame. P frames are encoded using both motion estimation and motion compensation. The number of frames between a P frame and the immediately preceding I frame is called the prediction span $M = 1-3$. Motion estimation compares small segments of two consecutive frames for differences. If a difference is detected, a search is carried out to determine the neighboring segment to which the original segment has moved. The search is limited to a few neighboring segments to reduce the search time. In applications involving very fast moving objects, B frames use search regions in both past and future frames. B frames provide highest level of compression and they do not propagate errors. The information in the I frames are decoded immediately as the frames are received. With P and B frames, correlation with preceding and or succeeding frames is necessary A fourth type of frames (PB frames) involves treating PB as one unit to increase the frame rate without increasing much the bit rate.

Using $4:1:1$, the Y matrix is 16×16 and known as macro block, while C_b *and* C_r matrices are 8×8 pixel each (DCT block size). To encode a P frame (target frame) contents of each macro block are compared on a pixel to pixel basis with the contents of the corresponding macro block in the preceding I or P frame (reference frame). Only the address of the macro block is encoded. Otherwise, the search is extended to cover an area around the macro block in the reference frame (Fig. 17.20).

Normally, only the contents of the Y matrix are used in the search, and if a match is found, two parameters are encoded, the motion vector - which is the x, y offset of the moving object - and the prediction error vector containing the difference values for $Y, C_b\ C_r$ (3matrices) in all locations. A moving object covers more than a single macro block. The motion vectors are encoded using differential coding and the resulting codewords are Huffman encoded. To encode a B frame, any motion is estimated with reference to I and P frames (preceding and succeeding) (Fig. 17.21).

The motion vector and the different matrices are computed using the preceding frame as the reference, and then the succeeding frame as a second reference. A new motion vector and set of different matrices are then computed using the target and the mean of the two predicted sets of values. The set with the lowest set of difference matrices is then chosen and encoded as a P frame. The motion vector is said to be a resolution of a fraction of a pixel (half pixel resolution).

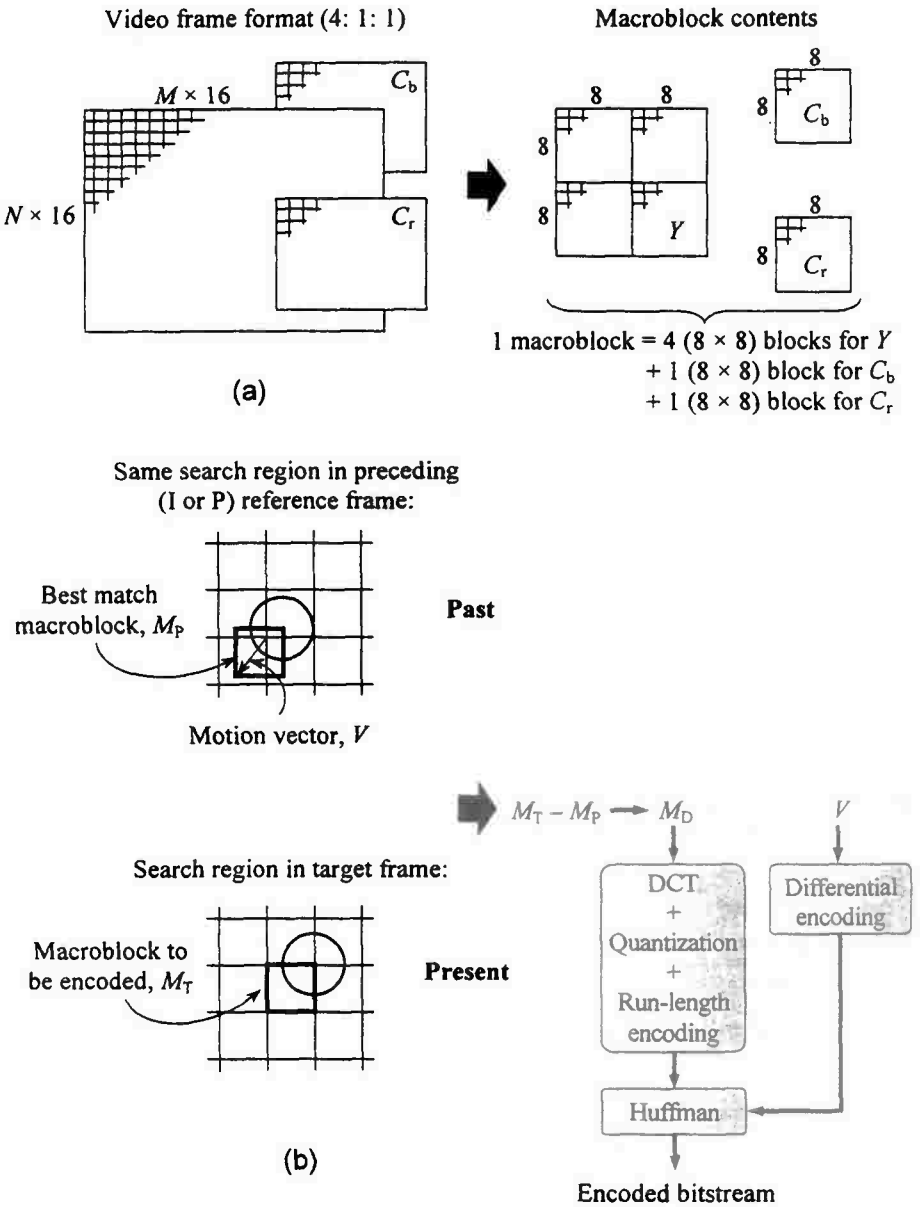


Fig. (17.20) *P* frame encoding
a) macro block b) encoding procedure

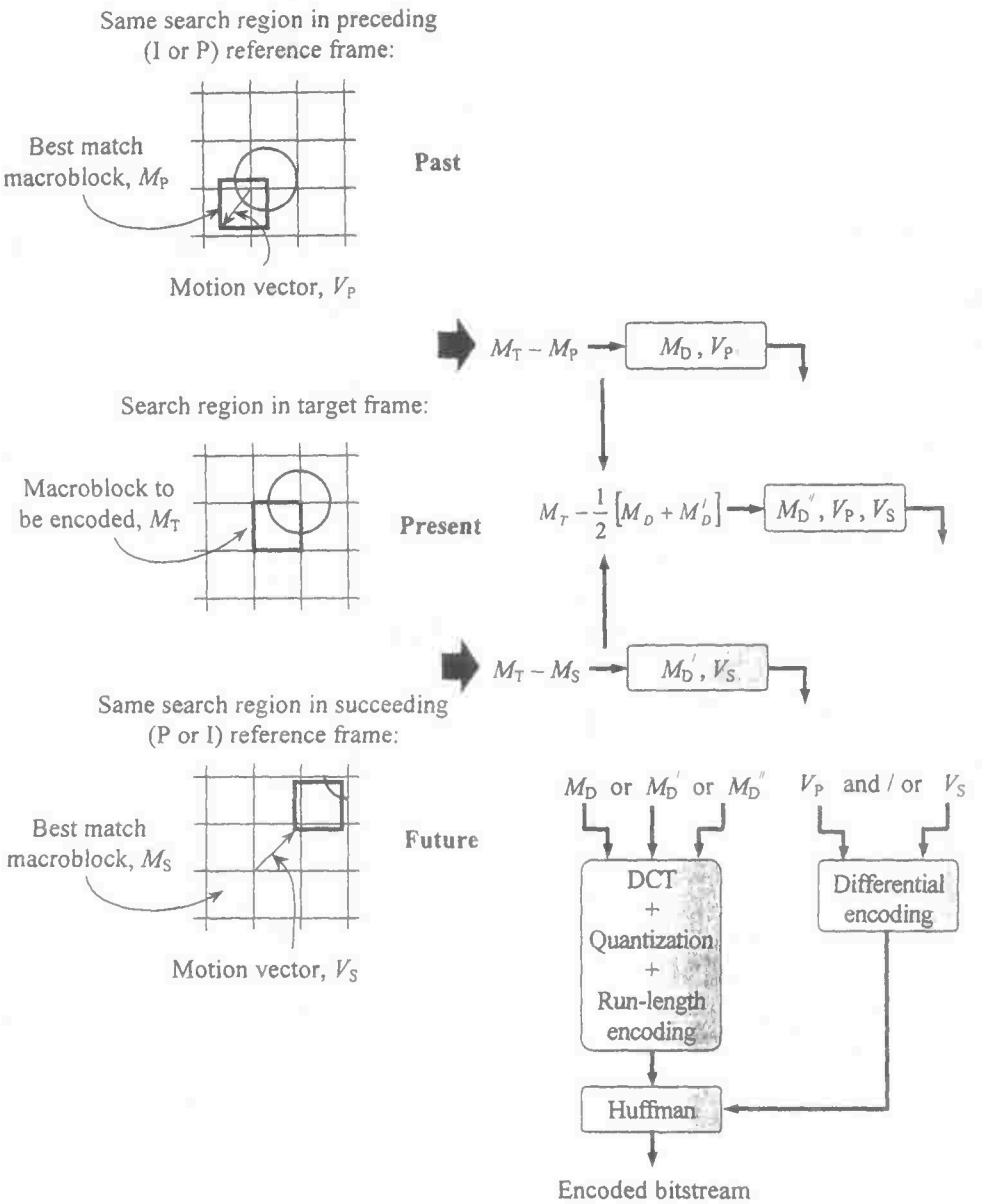


Fig. (17.21) *B* frame encoding
a) macro block b) encoding procedure

17.7 MPEG Standards:

The Motion Pictures Expert Group (MPEG) produced a set of standards relating to a range of multimedia applications that involve the use of video and audio based on the previous principles as follows:

- **MPEG1** is based on SIF digitization format with resolution up to 352×288 pixels. This standard is used for the storage of VHS quality audio and video on CD – Rom at bit rates up to $1.5\,Mbps$.
- **MPEG2** is intended for the reading and transmission of studio quality audio and video. It covers different levels of video resolution.
  - Low based on SIF digitization frames with resolution up to 352×288. It is compatible with **MPEG1,** with target bit rates up to $4\,Mpbs$.
  - Mean based on $4:2:0$ digitization format with a resolution up to 720×576 pixel. This produce studio quality digital quality. The target bit rate is $15Mpbs\ or\ 20Mpbs\ with\ 4:2:2$ format.
  - High based on $4:2:0$ format with a resolution 1440×1152 pixel. It is intended for high definition television (HDTV) with bit rates up to $60Mpbs$ or $80Mpbs$ with $4:2:2$ format.
  - High based on $4:2:0$ format with a resolution of 1920×1152 pixel. It is intended for video screen HDTV at a bit rate of up to $80Mpbs$ or $100Mpbs$ with $4:2:2$ format.
- **MPEG3** was focused an HDTV.
- **MPEG4** has very low bit rate channel 4:8:0 64 $kpbs$ intended for interactive multimedia applications over the internet and various types of entertainment networks.
- **MPEG7** is used in search engines to locate particular items of material with a defined feature.

Problems

1. For an image 8×8 the first row 4 pixels are bright and 4 pixels are dark. In the second row, the first 4 pixels are dark and the next 4 pixels are bright and so on. Obtain DCT.

2. Repeat the above problem for a checker board. What do you conclude?

3. Verify the quantized coefficients of (Fig. 17.7).

4. Assuming a quantization threshold value of 16, derive the resulting quantization error for each of the following DCT coefficients 127, 72, 64, 56, -64, -72, -128. Hence, find the maximum quantization error as a percent of the threshold value used.

5. Determine the encoded version of the following difference values in the DC coefficient from consecutive DCT blocks 12, 1, -2, 0, -1

6. Write down the runlength for the zigzag of Fig.(17.8).

7. Derive the binary form of the runlength encoded AC coefficients (0,6), (0,7), (3,3), (0,-1), (0,0).

8. In Prob. (17.6), determine the Huffman code for the DCT coefficients from consecutive DCT blocks, using the short default Huffman codewords (Table.17.2).

9. Derive the composite binary symbols for the following set of runlength encoded AC coefficients $(0,6),(0,7),(3,3),(0,-1),(0,0)$ using Table (17.3). Then write down the Huffman encoded bit stream.

10. Following on the JPEG encoder and decoder schematic, show the various computations carried out on a 80×80 pixels image.

References

1. "Multimedia Communications", F. Halsall, Addison. wesly, Harlow England, 2001.

2. "Image and Video Compression Standards, Algorithms and Architectures',' V. Bhaskaron, and K. Konstantinides, Academic press, N. Y., 1995.

3. "Digital Video and Audio compression", S. Solaris, Mc Graw Hill, N.Y.,1997.

4. Digital Compression of Still Images and Video", R. clarke, Academic press, N.Y., 1995.

5. " Digital Telephony " J. Bellamy, 2nd ed, J . Wiley, N.Y.,1991.