CHAPTER 7

Image Compression

Data is not information, Information is not knowledge, Knowledge is not understanding, Understanding is not wisdom.

–Cliff Stoll and Gary Schubert

LEARNING OBJECTIVES

Image compression is an important concept in image processing. Images and videos take a lot of space and their transmission takes time, given the bandwidth constraints. Therefore, image compression is necessary for both image storage and transmission. This chapter provides an overview of compression algorithms and standards. After studying this chapter, the reader will become familiar with the following:

- Image compression model
- Redundancy of data and its types
- Lossless compression algorithms
- Lossy compression algorithms
- Image compression standards

7.1 IMAGE COMPRESSION MODEL

Images require a lot of space as image files can be very large. They also need to be exchanged among various imaging systems. Hence, there is a need to reduce both the amount of storage space and the transmission time. This leads us to the area of data compression. Data compression deals with algorithms and techniques for compression of data. It is the art and science of removing redundancies to represent information in a compact form.

Data and information are two different things. Data is raw facts. Data is processed to give useful information. It has been observed that in data compression there is no compromise on the information quality; only the data used to represent the information is reduced.

Types of data

The types of data that are encountered in image processing can be called multimedia data and include the following:

1. Text data: This includes data present in flat files, which can be read and understood by human beings.
2. Binary data: This includes data that only machines can interpret, such as the data in executable files and metadata present in database files. Humans cannot interpret this data directly.
3. Image data: This is the pixel data that contains the intensity and colour information of the image.
4. Graphics data: This represents data in vector form.
5. Sound data: This data represents audio information.
6. Video data: This data represents video information.

Image processing applications use huge amounts of image data. They are expected to both store and transmit huge amounts of data. Data compression becomes essential due to the following three reasons:

Storage The storage requirements of imaging applications are very high. The goal of data compression is to reduce the amount of memory by reducing the number of bits, while at the same time maintain the minimum data to reconstruct the image. The reduction of data reduces the memory requirement and hence the resources spent for storage.

Transmission The transmission time of the image is directly proportional to the size of the image. Image compression aims to reduce the transmission time by reducing the size of the image. The reduction of data leads to easier and faster transportation of data.

Faster computation Reduced data often simplifies the algorithm design and facilitates faster execution of the algorithms.

Example 7.1 An image scan centre needs to store medical images whose resolution is $1024 \times 1024 \times 24$ bits. A total of 10,000 images are present. How much storage and transmission time will they require at 64 kbps?

Solution Number of bytes = $1024 \times 1024 \times 3$ bytes = 31,45,728 bytes

= 3145.728 Kb for one image

The total number of images in the scan centre is 10,000. Thus the scan centre's requirement is

= $10{,}000 \times 3145.728$ Kb

= 31,45,728 Kb

The compression scheme is as shown in Fig. 7.1.

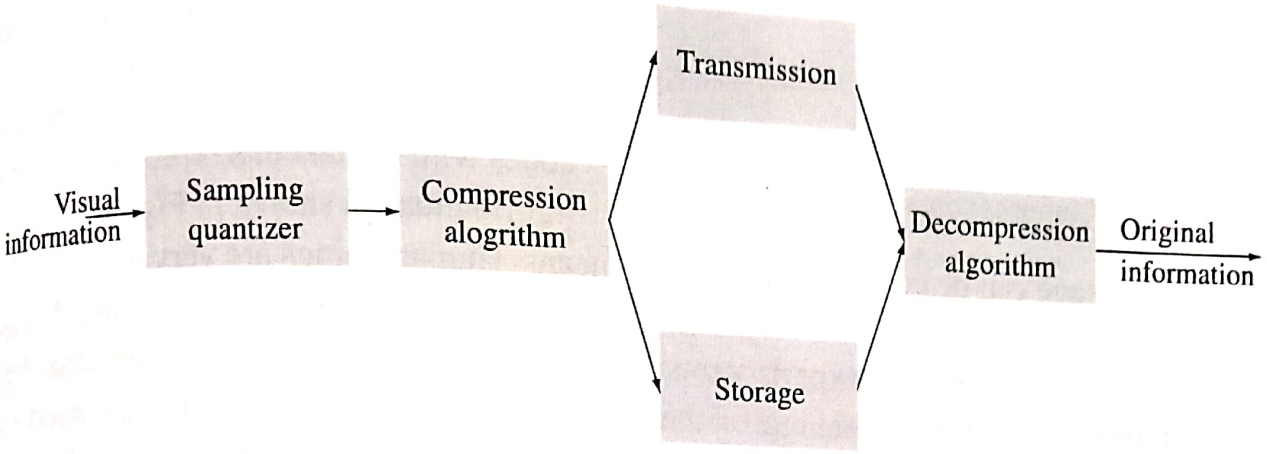


Fig. 7.1 Compression scheme

Compression and decompression algorithms in the form of a compressor and decompressor, respectively, may be located in the same place or at both ends of the channel. The compressor and decompressor are known as coder and decoder, respectively. The term encoder is also used for coder. The coder and decoder are collectively referred to as codec. Codec may be the hardware or the software component. Compression and decompression algorithms lay out the procedure or logic used for compression and decompression.

Figure 7.2 illustrates a simple image compression model. The two main components of the image compression model are the encoder and the decoder. The source or symbol encoder takes a set of symbols from the input data, removes the redundancies, and sends the data across the channel. The decoder has two parts, namely, channel decoder and symbol (or source) decoder. If the link is noise-free, the channel encoder and decoder can be omitted.

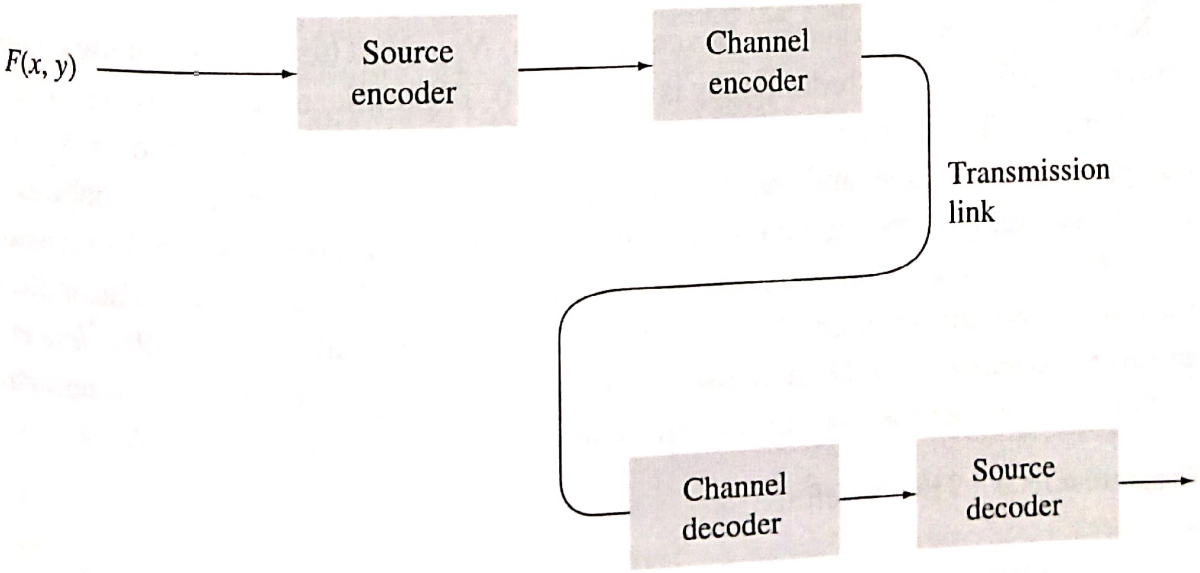


Fig. 7.2 Image compression model

7.1.1 Compression Measures

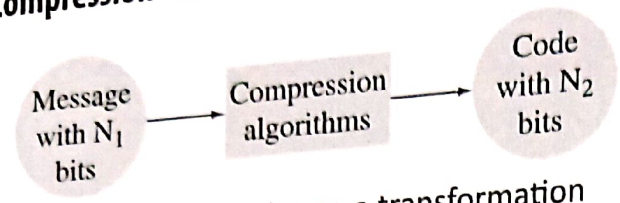


Fig. 7.3 Compression as a transformation

Data compression algorithms can be viewed as a mathematical transformation for mapping a message of N_1 data bits to a set of codes with N_2 data bits representing the same information, as shown in Fig. 7.3.

A message can be conveyed by various means. Human beings are very good at this. For example, we use abbreviations such as 'UK' and 'JPEG' to represent 'United Kingdom' and 'Joint photographic experts group', respectively. It can be observed that there is no compromise on the meaning of the message. Instead, only the representation of the message is changed so that it is more compact than the earlier form. This kind of logical substitution is called *logical compression*. This is used at the highest logical level. At the image level, the transformation of the input message to a compact representation of code is more complex. This is called *physical compression*, and is accomplished using complex data compression algorithms by manipulating the pixels.

In summary, data compression algorithms can be visualized as the mapping of a set of message symbols to codes using some logical rules of conversion. What is a message composed of? It is composed of a well-defined set of symbols S, which conveys something to the observer. In the imaging context, the pixel or block of pixels is considered as a set of symbols. A code, on the other hand, is a sequence of symbols or numbers that are used to represent information. A string of codes is called a codeword. The whole data compression process can now be visualized as a mapping of all possible sequences of symbols of a message (of N_1 bits), separately or in a file, to a set of codes, separately or in file, using N_2 bits. The compression ratio (C_R) is then defined as N_1/N_2 and the relative redundancy is defined as $R_D = 1 - \frac{1}{C_R}$.

Now three scenarios emerge. In scenario one, $N_2 = N_1$. This means that the compression ratio is 1 and the relative redundancy is $1 - 1/1 = 0$. This indicates that there is no redundancy in the image. The input message is reproduced exactly. In scenario two, if $N_2 << N_1$, the compression ratio is ∞ and the relative redundancy is 1. This condition implies that the image has highly redundant data and there is a significant compression. In scenario three, when $N_2 >> N_1$, the compression ratio is 0 and relative redundancy is ∞. This indicates that the transformed set has more data than the original set. Typically this is called *data explosion* or *reverse compression*. Data explosion is not under the purview of data compression.

Some of the metrics used to quantify compression measures are as follows:

Compression ratio This is defined as

$$\text{Compression ratio} = \frac{\text{Message (or file) size before compression}}{\text{Code (or file) size after compression}} = \frac{N_1}{N_2}$$

This is expressed explicitly as $N_1 : N_2$. It is common to use the compression ratio of 4:1. The interpretation is that 4 pixels of the input image are expressed as 1 pixel in the output image.

Savings percentage This is defined as

$$\text{Savings percentage} = 1 - \frac{\text{Message(or file) size after compression}}{\text{Code(or file) size before compression}} = 1 - \frac{N_2}{N_1}$$

Bit rate Bit rate describes the rate at which bits are transferred from the sender to the receiver and indicates the efficiency of the compression algorithm by specifying how much data is transmitted in a given amount of time. It is often given as bits per second (bps), kilobits per second (Kbps), or megabits per second (Mbps). Bit rate specifies the average number of bits per stored pixel of the image and is given as

$$\text{Bit rate} = \frac{\text{Size of the compressed file}}{\text{Total number of pixels in the image}} = \frac{N_2}{N} \text{ (bits per pixel)}$$

Example 7.2 An image is 8 MB before compression and 2 MB after compression. What are the values of compression ratio and savings percentage?

Solution The calculations are as follows:
Compression ratio is 8/2 = 4/1, that is, 4 : 1
Savings percentage is 1 – 2/8 = 6/8 = 3/4 = 75%.

7.2 COMPRESSION ALGORITHM AND ITS TYPES

The role of the compression algorithm is to reduce the source data to a compressed form and decompress it to get the original data. Hence the compression algorithm should have an idea about the symbols to be coded and their probability distribution.

Any compression algorithm has two components:

Modeller The purpose of the modeller is to condition the image data for compression using the knowledge of the data. The modeller is present both in the sender and the receiver sides. The models can be either static (that is, the models at the sender and the receiver sides do not change) or dynamic (that is, the models change depending on the change of data during the compression or decompression process). Based on the models used, the algorithm can also be classified as static or dynamic compression algorithm.

Coder The second component is called the coder. The sender-side coder is called the encoder. This codes the symbols independently or using the model. The receiver-side coder is called the decoder, which decodes the message from the compressed data.

If the models at the sender and receiver sides are the same, the compression scheme is symmetric. Otherwise, it is asymmetric.

Compression algorithms can be broadly classified into two types. Figure 7.4 illustrates the types of the compression algorithms.

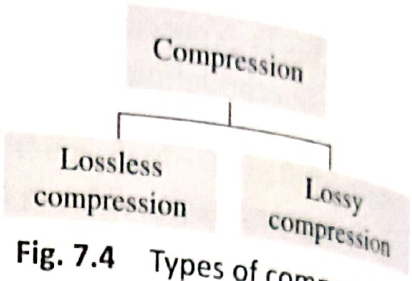


Fig. 7.4 Types of compression algorithms

Lossless compression is useful in preserving information as there is no information loss. This type of algorithm is useful in the legal and medical domains. Lossy compression algorithms compress the data with a certain amount of error that is acceptable to the human observer. The human visual system has many defects like colour blindness. So the loss of information is either not noticed much or the errors are acceptable. This category of algorithms is useful in applications such as broadcast television, and multimedia. The differences between lossless and lossy compression are listed in Table 7.1.

Table 7.1 Difference between lossless and lossy compression

Lossless compression	Lossy compression
This is a reversible process and no information is lost.	This is a non-reversible process and information is lost.
Compression ratio is usually less.	Compression ratio is very high.
It is used for data that humans can handle directly like text data. Compression is independent of the psychovisual system.	It is useful for diffused data that humans cannot understand or interpret directly. Compression is dependent on the psychovisual characteristics.
It is required in domains where reliability is very crucial such as executable files and medical data.	It is useful in domains where loss of data is acceptable.

Another way of classifying image compression algorithms is as follows:

1. Entropy encoding
2. Predictive coding
3. Transform coding
4. Layered coding

.1 Entropy Coding

The logic behind this category is that if pixels are not uniformly distributed, then an appropriate coding scheme can be selected that can encode the information so that the average number of bits is less than the entropy. Entropy (discussed in Chapter 3) specifies the minimum number of bits used to encode information. Hence the coding is based on the entropy of the source and on the possibility of occurrence of the symbols. This leads to the idea of variable length coding. Some examples of this type of coding are Huffman coding, arithmetic coding, and dictionary-based coding.

7.2.2 Predictive Coding

The idea behind predictive coding is to remove the mutual dependency between the successive pixels and then perform the encoding. Normally the samples would be very large, but the differences would be small.

For example, let us assume that the following pixels need to be transmitted:

Pixels	400	405	420	425
Difference		5	15	5

It can be observed that the difference of pixels is always lesser than original and requires fewer bits for representation. Therefore it makes sense to encode the differences rather than original. However, this approach may not work effectively for rapidly changing data such as {300, 4096, 128, 4096, 15}.

Hence the number of bits that is required to code the difference is very small for slowly varying data. Examples of this category include differential pulse code modulation (DPCM) and delta modulation techniques.

7.2.3 Transform Coding

The idea behind transform coding is to exploit the information-packing capability of the transform. The energy is packed into fewer components and only these components are encoded and transmitted. The human eye is more sensitive to the lower spatial frequencies than the higher spatial frequencies. The idea is to remove the redundant high frequency components to create compression. The removal of these frequency components leads to loss of information. However, this loss of information, if tolerable, can be used for imaging and video applications. Thus the basis of transform coding is frequency selection, information packing, and the concept of basis images.

7.2.4 Layered Coding

Layered coding is very useful in the case of layered images. Sometimes the image is represented in the form of layers. Data structures like pyramids are useful to represent an image in this multiresolution form. The layers of a pyramid would be sent depending on the application. At times, these images are segmented as foreground and background and based on the needs of the application, encoding is performed. This is also in the form of selected frequency coefficients or selected bits of pixels of an image.

7.3 TYPES OF REDUNDANCY

Redundancy means repetitive data. This may be data that share some common characteristics or overlapped information. This redundancy may be present implicitly or explicitly.

For example, consider the string *aaaaab*.

This information has redundancy where the character *a* is repeated five times. The message can be conveyed simply as $\{(a, 5), b\}$, implying that the character *a* occurs five times and *b* occurs once. This reduces the information to a compact form. This is useful when the repetition is large.

Similarly, redundancy can be present in images also. Consider the image shown in Fig. 7.5.

$$\begin{pmatrix} 10 & 10 & 10 \\ 10 & 10 & 10 \\ 10 & 5 & 2 \end{pmatrix}$$

Fig. 7.5 Image with redundant data

It can be observed that the pixel value 10 is repeated seven times. In this case, the redundancy is explicit. The redundancy may be implicit also. Consider the image shown in Fig. 7.6(a). This image can be split into two images by combining the LSBs to form one image and the MSBs to form another image, as shown in Fig. 7.6(b).

$$I = \begin{pmatrix} 00 & 01 \\ 00 & 01 \end{pmatrix} \qquad I = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \text{and} \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}$$

(a) (b)

Fig. 7.6 Implicit redundancy (a) Original image with implicit redundancy (b) Split images showing redundancy

In Fig. 7.6(b), the first image has all 0s and the second image has two 0s and two 1s. These redundancies can be exploited and the image data can be reduced. The types of redundancies in images are shown in Fig. 7.7.

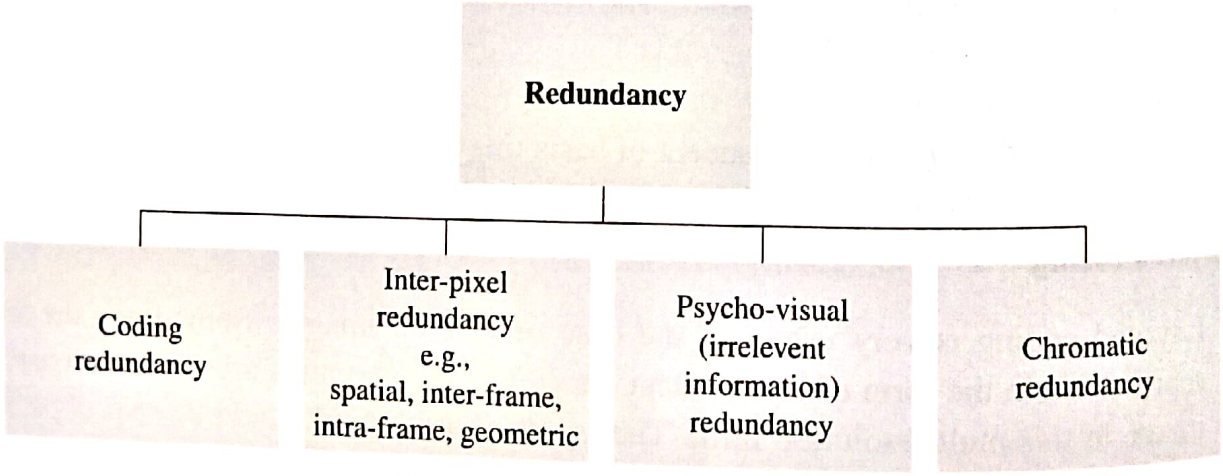


Fig. 7.7 Types of redundancy

Sections 7.3.1–7.3.4 discuss these redundancy types in detail.

.3.1 Coding Redundancy

Information theory uses probability to investigate information. It aims to measure information using the element of surprise, that is, the probability, associated with it. High

probabilities are associated with events that are likely to occur frequently. Low probabilities are associated with events that occur rarely. Based on this uncertainty, information can be measured and quantified.

The amount of uncertainty is called the self-information associated with the event. This is known as information content $I(S_i) = \log_2 \frac{1}{p_i}$ or $I(S_i) = -\log_2 p_i$. The unit is bits. Thus the information content is inversely proportional to its probability. When the probability of the event is 1, the information content is 0 and when the probability is 0, the information content is 1. Thus the range of information content is also 0–1.

In images, the grey values of pixels are not equally probable and often a value is related to that of its neighbours. So a pixel and its grey level (k) are modelled as a random variable r_k. Each random variable is associated with a probability $p(r_k)$. Coding redundancy is caused due to poor selection of coding technique. A coding scheme, as discussed earlier, assigns a unique code for all symbols of the message. For example, the message '4' can be coded as 100. Then the bit stream '100' is sent across the transmission channel and in the receiver side, it is decoded as message '4'. Obviously, binary coding is used here to code the message. Binary coding is a good example of a coding scheme where only two code symbols {0, 1} are used. So the message is mapped to a codeword containing 0s and 1s. There are many coding schemes available. ASCII (American standard code for information interchange) is a 7-bit code. Grey code is another popular code. When the mapping between the source symbols and the code is fixed, such a code is called a block code. If a code is uniquely decodable, it is called a uniquely decodable code. If the codeword of a block code is distinct, it is called a non-singular code. If the decoding is possible without the knowledge of the succeeding code words, it is known as instantaneous code.

Example 7.3 Calculate the entropy for the symbols shown in Table 7.2.

Table 7.2 Symbols and their distribution

Symbol	1	2	3	4	5	6
Probability	0.4	0.2	0.2	0.1	0.05	0.05

Solution Entropy $= -\Sigma p_i \times \log_2 p_i$; as $\log_z x = \log_{10} x / \log_{10} 2$

$= -[0.4 \times (\log10(0.4)/\log10(2)) + 0.2 \times (\log10(0.2)/\log10(2)) + 0.2 \times (\log10(0.2)/\log10(2)) + 0.1 \times (\log10(0.1)/\log10(2)) + 0.05 \times (\log10(0.05)/\log10(2)) + 0.05 \times (\log10(0.05)/\log10(2))]$

$= -[-0.5288 - 0.4644 - 0.4644 - 0.3322 - 0.2161 - 0.2161]$

$= 2.22$

It is observed that if binary code if used to code six symbols, at least 3 binary bits are required to encode them. It can be proved for most of the cases that variable length codes can be used in place of fixed sized codes like binary codes. The idea behind variable length

coding is to use shorter codes for the symbols that occur frequently and longer codes for the symbols that occur rarely. However, the Huffman code (explained in Section 7.4.2), a variable coding technique, uses lesser number of bits to encode the same information. Hence it can be concluded that variable length coding is better than fixed binary coding. Thus a wrong choice of code creates unnecessary additional bits. These extra bits are called redundancy. Thus coding redundancy is given as follows

$$\text{Coding redundancy} = \text{Average bits used to code} - \text{Entropy}$$

The average number of bits used to represent the message is given as

$$L_{avg} = \sum_{k=0}^{n} \ell(r_k).p(r_k)$$

Where $p(r_k)$ is the probability of the pixels given by grey levels r_k and $\ell(r_k)$ is the length of the code used. Hence the total number of bits required to code an image of size $M \times N$ is given as $M \times N \times L_{avg}$. Entropy of an image, as discussed in Chapter 3 is given as $H = -\sum_{i=1}^{n} p_i \log_2(p_i)$ and denotes the minimum number of bits required to code the message. It can be observed that if the average length of the code equals entropy, the coding redundancy is zero.

Thus coding redundancy is removed by using good coding schemes including variable length codes such as Huffman coding and Shannon-Fano coding algorithms discussed in section 7.4.2 and 7.4.3, respectively.

Example 7.4 Calculate the coding redundancy for the symbols given in Example 7.3 whose codes are shown in Table 7.3.

Table 7.3 Symbols and their variable length code

Symbol	1	2	3	4	5	6
Huffman code	0	10	110	1110	11110	11111
Binary code	000	001	010	011	100	101

Solution Entropy = 2.22 (from Example 7.3)

Average length for binary code = 3

Therefore coding redundancy of binary coding is 3 – 2.22 = 0.78

Average length for Huffman code = $\Sigma l_i \times p_i$

$$= (0.4 \times 1) + (0.2 \times 2) + (0.2 \times 3) + (0.1 \times 4) + (0.05 \times 5) + (0.05 \times 5) = 2.3$$

Therefore coding redundancy of Huffman coding is 2.3 – 2.22 = 0.08

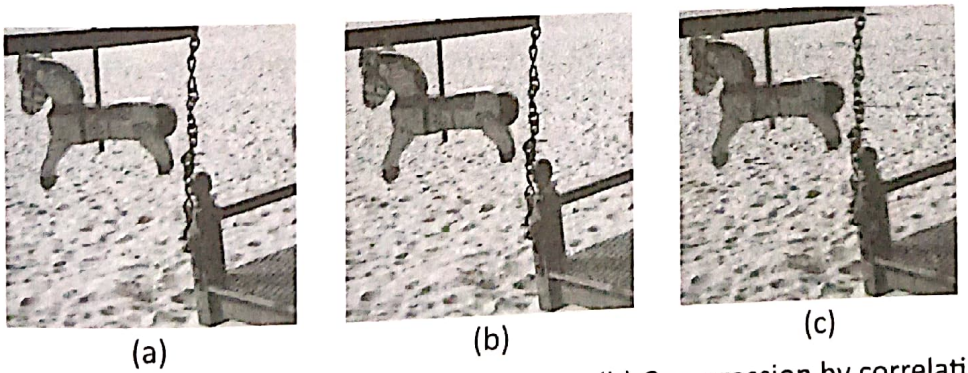
It can be observed that Huffman coding is closer to the entropy. Hence it can be concluded that variable length coding is better than fixed binary coding.

7.3.2 Inter-pixel Redundancy

A pixel of an image is not isolated, but often related to its neighbours. Hence, very often a pixel can be predicted using the value of its neighbours. For example, consider an image with a constant background. The visual nature of the image background is given by many pixels that are not actually necessary. This is called spatial redundancy (or geometrical redundancy). Spatial redundancy may be present in a single frame (intra-frame) or among multiple frames (inter-frame or temporal redundancy). The autocorrelation function can be used to study the spatial redundancy that is present in an image.

In intra-frame redundancy, large portions of the image may have the same characteristics such as colour and intensity. This kind of redundancy is also called as spatial redundancy. One way to reduce the redundancy is to use subsampling, where alternative pixels can be used to reduce the bits. The average of the group of pixels can also be used as a subsampling measure. Another way to reduce the inter-pixel dependency is to use quantization where a fixed number of bits are used to reduce the bits. Inter-pixel dependency is solved by algorithms such as predictive coding techniques, bit-plane algorithm, run length coding, and dictionary-based algorithms.

Consider the image shown in Fig. 7.8(a). It is compressed by JPEG at various quality levels, such as 50% and 95%, as shown in Figs 7.8(b) and 7.8(c), respectively. It can be observed that the picture quality is more or less the same even at 95% compression. This indicates the presence of highly redundant pixels in the image.



(a) (b) (c)

Fig. 7.8 Inter-frame redundancy (a) Original image (b) Compression by correlation at 50% (c) Compression by correlation at 95%

7.3.3 Psychovisual Redundancy

Most imaging applications produce images that are observed by humans. So the images that convey little or no information to the human observer are said to be psychovisually redundant. For example, the human visual system is very sensitive to certain information

such as edges and textures. In most cases, the outline of the object itself conveys the structure of the object. So, the human visual system attaches great importance to these kinds of information compared to any other information of the image. This kind of redundant information of less importance is called psychovisual redundancy. One way to resolve this redundancy is to perform uniform quantization by reducing the number of bits. The least significant bits (LSBs) of the image do not convey much information and hence can be removed. Sometimes this may cause edge effects that can be removed by a scheme called *improved gray scale* (IGS). This is a coding scheme where a set of least significant bits of a pixel is removed. Then a random number based on the four LSBs of a pixel and its previous pixel is generated and added with the current pixel value. This is to avoid edge effects and then the modified pixel values are used for display. However, if the pixel is of the form 1111 xxxx, then to avoid overflow 0000 is added. An original image and its IGS-applied versions are shown in Figs 7.9(a)–7.9(c).

(a) (b) (c)

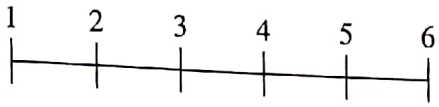
Fig. 7.9 Psychovisual redundancy (a) Original image—24 bits (b) IGS—8 bits (c) IGS—16 bits

It can be observed that the picture quality at 16 bits is almost similar to the original image.

7.3.4 Chromatic Redundancy

Chromatic redundancy refers to the presence of unnecessary colours in an image. The colour channels of colour images are highly correlated and the human visual system cannot perceive millions of colours. Hence, the colours that are not perceived by the human visual system can be removed without affecting the quality of the image.

Fidelity refers to the condition of accurate reproduction of data. The difference between the original and the reconstructed images is called distortion. The amount of distortion should be assessed. Objective fidelity measures such as error, SNR, and PSNR, for assessing the quality of an image have been discussed in Chapter 5. Subjective fidelity measures are based on the view that judgments of the image quality are often subjective. Subjective assessment can be compared to the process of buying of a television set in a shop where many television sets are available. Rather than bothering about the objective measures involving mathematical formulae, we often use some subjective assessment factors such as picture quality, appearance, brand name, or customer care to select the television set. Similarly the image quality can be assessed based on the subjective picture quality scale (PQS) which is as follows.



The scale indicates values in the range 1–6. The value 1 indicates that the image is unusable, 2 indicates an inferior quality image, 3 indicates the presence of interference, 4 indicates an acceptable image, 5 indicates a good image, and 6 indicates an extremely high quality image. Subjective measures can be made more effective by taking a group of observers instead of a single observer. The group index can then be calculated and used.

7.4 LOSSLESS COMPRESSION ALGORITHMS

Lossless compression algorithms preserve information and the compression process incurs no data loss. Hence these algorithms are used in domains where reliability and preservation of data are crucial. However, the compression ratio of these algorithms is small in comparison to the lossy compression algorithms. Some popular lossless compression algorithms are as follows:

1. Run-length coding
2. Huffman coding
3. Shannon–Fano coding
4. Arithmetic coding
5. Dictionary-based coding

Sections 7.4.1–7.4.7 discuss the design and implementation of lossless algorithms.

7.4.1 Run-length Coding

Run-length coding (RLC) exploits the repetitive nature of the image. It tries to identify the length of the pixel values and encodes the image in the form of a run. Each row of the image is written as a sequence. Then the length is represented as a run of black or white pixels. This is called run-length coding. This is an effective way of compressing an image. If necessary, there can be further compression using variable length coding to code the run lengths themselves.

Run-length coding is a Consultative Committee of the International Telegraph and Telephone (CCITT), a new standard that is used to encode binary and grey-level images. The technique scans the image row by row and identifies the run. The output run-length vector specifies the pixel value and the length of the run.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Fig. 7.10 Sample binary image for run-length coding

Consider the image shown in Fig. 7.10.

The horizontal RLC starts from the top–left pixel, scans the image from left to right, and generates the run-length vector. For the image in Fig. 7.10, the run-length vectors are as follows:

(0, 5)
(0, 3), (1, 2)
(1, 5)
(1, 5)
(1, 5)

The maximum length here is five. It requires three bits in binary. The total number of vectors is six. The maximum number of bits required is three. The number of bits per pixel is one, as the pixels of the image are either 0 or 1. Therefore, the total number of pixels is given by $6 \times (3 + 1) = 24$. The total number of bits of the original image is $5 \times 5 = 25$. Therefore, the compression ratio is 25/24, that is, 1.042:1.

The scan line can be changed. This change affects the compression ratio. Vertical line scanning of the same image yields

(0, 2) (1, 3)
(0, 2) (1, 3)
(1, 2) (1, 3)
(0, 1) (1, 4)
(0, 1) (1, 4)

Here the total number of vectors = 10
Maximum number of bits = 3
Number of bits per pixel = 1
Therefore, $10 \times (3 + 1) = 40$
Compression ratio = 25/40 = 0.625:1

It can be observed that this is significantly lesser than the previous scheme.
The scan line can be changed to a zigzag line as shown in Fig. 7.11.
Vertical line scanning yields

(0, 5)
(1, 2) (0, 3)
(1, 5)
(1, 5)
(1, 5)

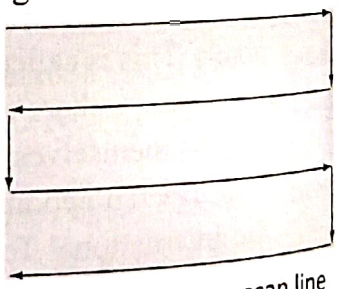


Fig. 7.11 Zigzag scan line

The total number of pixels here is $6 \times (3 + 1) = 24$
This yields a compression ratio of 25/24 = 1.041
It can be observed that the compression ratio changes with the scan line.
Theoretically, if

1. the estimate of the entropy of the black run is H_0,
2. the estimate of the entropy of the white run is H_1,
3. the estimate of the average value of the black run is L_0, and
4. the estimate of the average value of the white run is L_1,

the approximate run-length entropy of the image can be given as

$$H_{\text{run-length}} = \frac{H_0 + H_1}{L_0 + L_1}$$

7.4.1.1 Two-dimensional RLC

A facsimile (fax) machine raster scans a document and creates an electrical signal with a strong impulse for dark spots and a weak impulse for bright spots. Fax machines use 2376 scan lines for a document and each scan line has 1728 dots. Hence a fax image is of size 2376×1728. The electrical signal is then digitized and sent across the telephone line. A fax machine uses lossless compression techniques for transmission.

CCITT standards include Group 1 (G1), Group 2 (G2), Group 3 (G3), and Group 4 (G4) standards for facsimile transmission. The groupings are based on the apparatus used. G1 and G2 standards are for analog fax devices, G3 and G4 are for digital transmission. G3 standard aims to transmit an A4-sized document in one minute. T.4 and T.6 are recommendations of G3 and G4 standards, respectively, for fax compression.

Let us consider a situation where all 64 pixels of an image are white. There is no need to code it as eight lines, instead it can be coded effectively using a procedure called modified Huffman (MH) code. This algorithm is useful for transmitting documents in facsimile.

Modified Huffman code uses two types of codes:

1. Termination codes—these codes govern the black and white run lengths from 0 to 63.
2. Make-up codes—these codes govern run lengths that are multiples of 64 pixels.

A sample portion of a termination code is shown in Table 7.4.

Table 7.4 Sample termination code

n	White runs	Black runs	*n*	White runs	Black runs
0	00110101	0000110111	32	00011011	000001101010
1	000111	010	33	00010010	000001101011
2	0111	11	34	00010011	000011010010
3	1000	10	35	00010100	000011010011

A sample portion of a make-up code is shown in Table 7.5.

Table 7.5 Sample make-up code

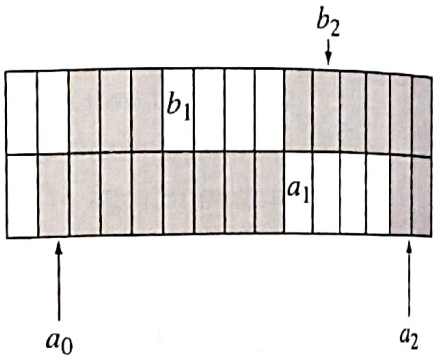
n	White runs	Black runs	*n*	White runs	Black runs
64	11011	0000001111	960	011010100	0000001110011
128	10010	000011001000	1024	011010101	0000001110100
192	010111	000011001001	1088	011010110	0000001110101

If the run length is less than 63, only termination codes are used. If the run length is more than 63, the MH algorithm represents it as

$$\text{Run length} = (M \times 64) + T$$

M is an integer whose value ranges is 1–27 and $(M \times 64)$ is the make-up code and T is the termination code.

One-dimensional RLC can be extended to 2D also. CCITT T.4 recommendation implements an algorithm called READ (relative element absolute designate). This is also known as the Modified READ (MR) algorithm. This algorithm is based on the relative positions of pixel changes. If a 2D image needs to be coded, then a single line is chosen for encoding. This is called the *coding line*. The coding is done with respect to another line called the *reference line*. The idea is to look for blocks of codes that can be represented effectively. This is done by identifying five changing pixels a_0, a_1, a_2 in the coding line, and b_1 and b_2 in the reference line.



Consider the code on the right. The pixels a_0, a_1, and a_2 represent the colour changes of the coding line. a_0 and a_2 share the same colour change of white to black and a_1 represents the opposite colour change of black to white. b_1 and b_2 are changing pixels of the reference line. b_1 has the same colour of a_1 and b_2 represents the next colour change in the reference line. Based on the relative positions of these five pixel changes, RLC is encoded for 2D images in three modes. The modes are called pass mode, horizontal mode, and vertical mode.

If b_2 is located on the left of a_1, then the condition is known as pass mode. In that case a special code '0001' is sent. If the distance between a_1 and b_1 differs by a maximum of ±3 pixels, then this scenario is called vertical mode. In that case the difference between the run lengths a_1 and b_1 is calculated and coded. If the distance is more than three pixels, this scenario is called horizontal mode and in that case, $001 + (a_0\ a_1) + (a_1\ a_2)$ is encoded and sent to the receiver.

The CCITT T.6 recommendation is similar to the earlier scheme in all aspects, but the main difference is that, only 2D coding is used here, that is, there is no 1D coding. In addition, the *k*-factor is absent in this scheme. Therefore there is no restriction placed on the coding scheme. This modifies the READ algorithm and this algorithm is known as modified modified READ algorithm (MMR). This standard also has the ability to handle coding lines of more than 2623 pixels.

Another useful standard is the Joint Bi-level Image processing Group (JBIG) proposed by the International Standards Organization (ISO), the International Electrochemical Commission (IEC), and the CCITT. The idea behind these standards is pattern matching techniques. Since the scanned text, unlike the written document, contains standard fonts for encoding the image, the patterns can be isolated and a library of patterns can

be formed. Then during the transmission, a close match for the given pattern is found, and its index from the library is sent across. JBIG T.82 standard uses two algorithms—progressive transmission algorithm and a lossless algorithm discussed in Section 7.6.1. JBIG2 is another standard for providing lossy compression of bi-level images. The recommendation divides the document into three portions—text, halftone, and, 'other region' which includes all other portions of the document. The text portion is encoded using a lossy or a lossless manner. The halftone is the image part. It is coded with the pattern matching approach. The other area is encoded with MMR algorithm and this approach is very effective.

7.4.2 Huffman Coding

Huffman coding is a type of variable length coding. In Huffman coding, the coding redundancy can be eliminated by choosing a better way of assigning the codes. The Huffman coding algorithm is given as follows:

1. List the symbols and sort them.
2. Pick two symbols having the least probabilities.
3. Create a new node. Add the probabilities of the symbols selected in step 2 and label the new node with it.
4. Repeat steps 2 and 3 till only one node remains.
5. Start assigning code 0 for the left tree and code 1 for the other branch.
6. Trace the code from the root to the leaf that represents each label.

The running time of the algorithm is $O(n\log n)$. The Huffman tree is shown in Fig. 7.12 in Example 7.5.

Example 7.5 Calculate the Huffman coding for the set of symbols shown in Table 7.6.

Table 7.6 Symbols and their probabilities

Symbol	A	B	C	D
Probability	0.4	0.3	0.2	0.1

Solution The Huffman tree is constructed as shown in Fig. 7.12. This yields the code shown in Table 7.7.

Table 7.7 Symbols and their Huffman codes—version 1

Symbol	A	B	C	D
Code	0	10	111	110

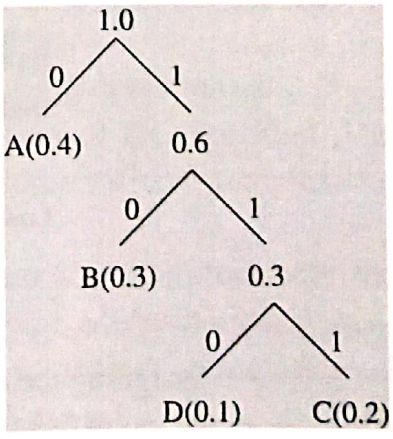


Fig. 7.12 Huffman code tree

The problem with Huffman codes is that they are not unique. The data given can be differently combined to yield the result shown in Fig. 7.13.

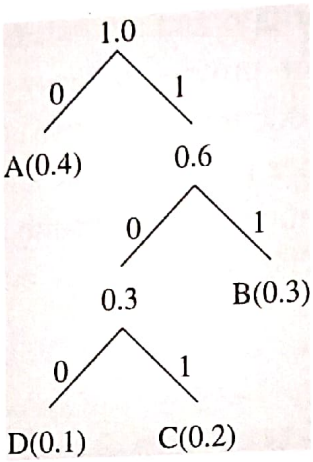


Fig. 7.13 Huffman code tree—alternate version

This yields the code shown in Table 7.8.

Table 7.8 Symbols and their Huffman codes—version 2

Symbol	A	B	C	D
Code	0	11	101	100

7.4.2.1 Canonical Huffman code

The canonical Huffman code is a variation of the Huffman code. A tree that is constructed using the following rules is called the canonical Huffman code tree:

1. The newly created item is given priority and placed at the highest position in the sorted list.
2. In the combination process, the higher-up symbol is assigned code 0 and the lower-down symbol is assigned code 1.

This is shown in Table. 7.9.

Table 7.9 Canonical Huffman code

Rank	Initial	Pass 1	Pass 2	Pass 3
Highest	A = 0.4	A = 0.4	BDC = 0.6	ABDC = 1.0
	B = 0.3	B = 0.3	A = 0.4	
Lowest	C = 0.2	DC = 0.3		
	D = 0.1			

Assigning the code as per the rules yields the code as shown in Table 7.10.

Table 7.10 Symbols and their canonical Huffman codes

Symbol	A	B	C	D
Code	1	00	010	011

7.4.2.2 Huffman decoder

The procedure for decoding as implemented in the Huffman decoder is as follows:

1. Find the coded message. Start from the root.
2. If the read bit is 0, move to the left. Otherwise, move to the right of the tree.
3. Repeat the steps until the leaf is reached. Then generate the code and start again from the root.
4. Repeat steps 1–3 till the end of the message.

7.4.2.3 Characteristics of Huffman coding

Huffman codes are also efficient. The coding efficiency of the Huffman code can be obtained using the formula

$$\text{Huffman code efficiency} = \frac{\text{Entropy}}{\text{Average length of the Huffman code used}}$$

Decoding problems arises only when one code is a prefix of another code. For example if the code for A is 1, B is 0, and C is 10, and if a stream '110' is available, then this may cause interpretation problems as the stream can be coded either as AAB or as AB. The main advantage of Huffman code is its unique decidability, that is, it does not produce a code that is prefix of another code.

One of the major disadvantages of Huffman code is that all the symbols along with their frequency count should be available beforehand. This is not possible when sending messages over the Internet as the message has to be sent in streams (since the complete message is not available). However, there are variants of Huffman codes, like the dynamic Huffman code, to tackle this dynamic environment. The difference between the static and dynamic Huffman codes is that static codes are fixed and the statistical characteristics of the data are known in advance, whereas dynamic codes are not fixed and the statistical characteristics are not fully known in advance.

7.4.2.4. Truncated Huffman coding

A Huffman code involving N source symbols requires $N-2$ sorting operations and $N-2$ coding assignments. This increases the computational complexity of the algorithm. The two variants of Huffman code which can be used for reducing computational complexities are truncated Huffman coding and shift Huffman coding.

Truncated Huffman coding is similar to the general Huffman algorithm, but only the most probable K items of N source symbols are joined together with a hypothetical symbol and are coded with the standard Huffman code. The rest of the symbols N–K are assigned the Huffman code of the hypothetical symbol concatenated with the binary code of length $\log(N-K)$. When $K = J$, this reduces to the standard Huffman coding procedure.

The procedure for truncated Huffman coding is given as follows:

1. Arrange all the N source symbols in monotonically decreasing order.
2. Divide the total number of symbols (N) into two parts. The first part is the set of most probable symbols K. Add the hypothetical symbol for that.
3. The second group consists of remaining $N - K$ symbols.
4. Arrange the new set of symbols monotonically decreasing in the sorted order so that the probability is decreasing.
5. Encode the new symbols in standard Huffman code.
6. The less probable symbols are assigned Huffman code concatenated with the binary code of length log $(N - K)$.

Example 7.6 Construct the truncated Huffman coding for the following set of symbols.

Source symbols	P_i
S1	0.28
S2	0.22
S3	0.2
S4	0.08
S5	0.07
S6	0.03
S7	0.02
S8	0.05
S9	0.05

Solution The most probable symbols are 3, so $K = 3$. All the remaining symbols (i.e., $9 - 3 = 6$ symbols) can be combined (Symbols S4 to S9) into a single hypothetical symbol. Apply Huffman code procedure by assuming the targest probability symbol on the left side of the tree during the encoding process. The resultant table in given here.

Source symbols	P_i	Truncated Huffman
Sx	0.3	1
S1	0.28	01
S2	0.22	000
S3	0.2	001

Now, the less probable symbols are assigned the code of hypothetical symbols concatenated with the binary code of length 3. This results in the final code as follows:

Source symbols	P_i	Truncated Huffman code
S1	0.28	01
S2	0.22	000
S3	0.2	001
S4	0.08	1 000

(Contd)

(*Contd*)

Source symbols	P_i	Truncated Huffman code
S5	0.07	1 001
S6	0.03	1 010
S7	0.02	1 011
S8	0.05	1 100
S9	0.05	1 101

This is the final truncated Huffman code. The efficiency of this code can be evaluated as

$$L_{avg} = \sum_{i=1}^{9} P_i \times L_i$$

$$= 0.28\times 2+0.22\times 3+0.2\times 3+0.08\times 4+0.07\times 4+0.03\times 4+0.02\times 4+0.05\times 4+0.05\times 4$$

$$= 3.02 \text{ bits/symbols}$$

Huffman shift code Huffman shift code is another variation of Huffman code. Huffman shift codes are suboptimal but computationally faster than the traditional Huffman algorithm. The Huffman shift code is given as follows:

1. Arrange the symbols in monotonically decreasing order.
2. Divide the number of symbols into equal-sized blocks.
3. All the symbols in a block are coded using the Huffman algorithm.
4. Distinguish each block with a special symbol, that is, the Huffman code of the hypothetied symbol.
5. The least frequent symbol *i* is assigned a code that is characterized by the Huffman code of the hypothetical symbol and *i* symbol code of the first block.

Using the block identification, the respective blocks can be accessed and by shifting it, the codes of the symbols can be obtained.

Example 7.7 Construct the Shift Huffman coding for the following set of symbols.

Source symbols	P_i
S1	0.28
S2	0.22
S3	0.2
S4	0.08
S5	0.07
S6	0.03
S7	0.02
S8	0.05
S9	0.05

Solution The most probable symbols are 3, so $K = 3$. All the remaining symbols (i.e., $9 - 3 = 6$) can be combined (Symbols S4 to S9) into a single hypothetical symbol. Apply Huffman code

procedure by assuming the largest probability symbol on the left side of the tree during the encoding process. The resultant table in given here:

Source Symbols	P_i	Truncated Huffman
Sx	0.3	1
S1	0.28	01
S2	0.22	000
S3	0.2	001

In shift Huffman coding, the code word for the less probable symbols is constructed by means of concatenating the special prefix code of the hypothetical symbol and code of the reference block. This results in the final code as follows:

Source Symbols	P_i	Huffman shift code
S1	0.28	01
S2	0.22	000
S3	0.2	001
S4	0.08	1 01
S5	0.07	1 111
S6	0.03	1 001
S7	0.02	11 01
S8	0.05	11 000
S9	0.05	11 001

This is the final shift Huffman code. The efficiency of this code can be evaluated as

$$L_{avg} = \sum_{i=1}^{9} P_i \times L_i$$
$$= 0.28\times2+0.22\times3+0.2\times3+0.08\times3+0.07\times4+0.03\times4+0.02\times4+0.05\times5+0.05\times5$$
$$= 3.04 \text{ bits/symbols}$$

7.4.3 Shannon–Fano Coding

The difference between Shannon–Fano coding and Huffman coding is that the binary tree construction is top-down in the former. The whole alphabet of symbols is present in the root. Then a node is split into two halves along with the probability of the symbols—one corresponding to the left and the other to the right, based on the values of the probabilities. This process is repeated recursively and an entire tree is constructed. Then 0 is assigned to the left half and 1 to the second half.

The steps of the Shannon–Fano algorithm are as follows:

1. List the frequency table and sort the table on the basis of increasing frequency.
2. Divide the table into two halves such that the groups have more or less equal number of frequencies.
3. Assign 0 to the upper half and 1 to the lower half.
4. Recursively repeat the process until each symbol becomes a leaf on the tree.

Example 7.8 Consider the symbols and frequency shown in Table 7.11. Construct the Shannon–Fano code.

Table 7.11 Symbols and their frequencies for Shannon–Fano code

Symbol	A	B	C	D	E
Frequency	12	8	7	6	5

Solution First division (Table 7.12):

Table 7.12 First division

Symbol	A	B	C	D	E
Frequency	12	8	7	6	5
Sum	(20)		(18)		
Assign bit	0		1		

Second division (Table 7.13):

Table 7.13 Second division

Symbol	A	B	C	D	E
Frequency	12	8	7	6	5
Sum	12	8	7	11	
Code	00	01	10	11	

Third division (Table 7.14):

Table 7.14 Third division

Symbol	A	B	C	D	E
Frequency				6	5
Sum				6	5
Code				110	111

The Shannon–Fano tree is shown in Fig. 7.14.

Assigning the code as per the rules yields the codes in Table 7.15.

Table 7.15 Symbols and their Shannon–Fano codes

Symbol	A	B	C	D	E
Code	00	01	10	110	111

The decoding process of the Shannon–Fano scheme is the same as that of the Huffman scheme, as both these codes are prefix codes.

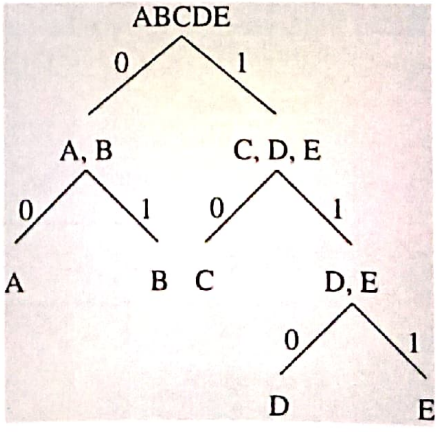


Fig. 7.14 Shannon–Fano code tree

Example 7.9 Construct Shannon–Fano code for the following set of symbols given with their probability.

Symbols	A	B	C	D	E	F	G
Probability	0.3	0.2	0.1	0.1	0.1	0.05	0.05

Solution First division

Symbols	**A**	**B**	**C**	**D**	**E**	**F**	**G**
Frequencies	0.3	0.2	0.1	0.1	0.1	0.05	0.05
Sum	0.5		0.4				
Assign List	0		1				

Second division

Symbols	**A**	**B**	**C**	**D**	**E**	**F**	**G**
Frequencies	0.3	0.2	0.1	0.1	0.1	0.05	0.05
Sum	0.3	0.2					
assign List	00	01					

Third division

Symbols	**A**	**B**	**C**	**D**	**E**	**F**	**G**
Frequencies			0.1	0.1	0.1	0.05	0.05
Sum			0.2		0.2		
Assign List					11		

Fourth division

Symbols	**A**	**B**	**C**	**D**	**E**	**F**	**G**
Frequencies			0.1	0.1	0.1	0.05	0.05
Sum							
code			100	101		111	

Fifth division

Symbols	**A**	**B**	**C**	**D**	**E**	**F**	**G**
Frequencies					0.1	0.05	0.05
Sum					0.1	0.1	
code					110	111	

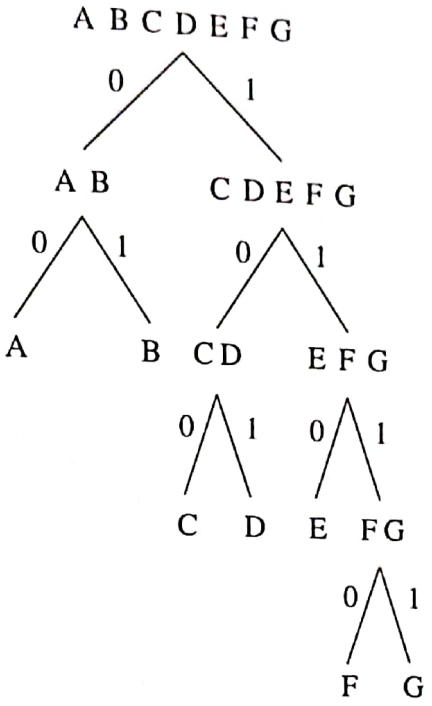
Sixth division

Symbols	**A**	**B**	**C**	**D**	**E**	**F**	**G**
Frequencies						0.05	0.05
Sum							
code						1110	1111

Finally we get

Symbols	A	B	C	D	E	F	G
Codes	000	010	100	101	110	1110	1111

In tree form, the aforementioned symbols are shown here:



7.4.4 Bit-plane Coding

The idea of RLC can also be extended for multilevel images. This technique splits a multilevel image into a series of bi-level images, that is, an m-bit grey level image can be represented in the form

$$a_{m-1}2^{m-1} + a_{m-2}2^{m-2} + \ldots + a_1 2^1 + a_0 2^0$$

The zeroth order bit plane is generated by collecting the a_0 bits of each pixel. The first order bit plane is generated by collecting all the first bits of each pixel. Continuing in this manner, the $m - 1$ order bit plane is generated by collecting all the a_{m-1} bits of each pixel.

Let us assume that the grey scale image is as follows

$$A = \begin{pmatrix} 2 & 6 & 6 \\ 6 & 7 & 7 \\ 1 & 2 & 4 \end{pmatrix}$$

The binary equivalent of this image is $A = \begin{pmatrix} 010 & 110 & 110 \\ 110 & 111 & 111 \\ 001 & 010 & 100 \end{pmatrix}$

The image A can now be divided into three planes using the MSB, the middle bit, and the LSB as follows:

$$A(\text{MSB}) = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}; A(\text{mid}) = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix}; A(\text{LSB}) = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

An 8-bit image and its individual planes are shown in Figs 7.15(a)–7.15(i).

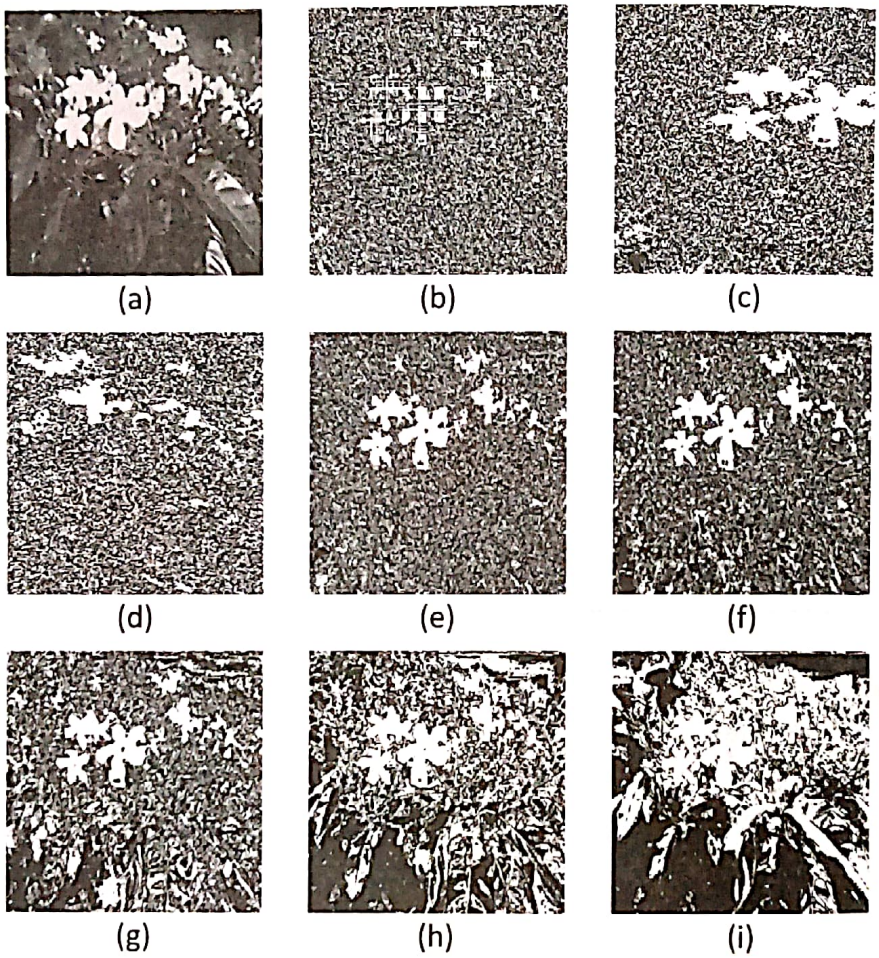


Fig. 7.15 Bit-plane coding (a) Original image (b) 0th bit plane (c) 1st bit plane (d) 2nd bit plane (e) 3rd bit plane (f) 4th bit plane (g) 5th bit plane (h) 6th bit plane (i) 7th bit plane

The individual planes of the image can now be compressed using RLC techniques. If a plane is completely white or black, RLC yields peak performance. However, the disadvantage of this scheme is that the neighbours in the spatial domain, say 3 and 4, having binary codes 011 and 100, are not present together in any of the planes. So it is reasonable to expect that the close neighbours in the spatial domain not being present together in the bit plane causes a problem. In addition, small changes in the grey level have a chain effect and the complexity of the bit plane is significantly affected.

To avoid this problem, grey code can be used instead of binary code. In grey code, the successive codes differ by only one bit. The algorithm for generating grey code can be given as follows:

The image A can now be divided into three planes using the MSB, the middle bit, and the LSB as follows:

$$A(\text{MSB}) = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}; A(\text{mid}) = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix}; A(\text{LSB}) = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

An 8-bit image and its individual planes are shown in Figs 7.15(a)–7.15(i).

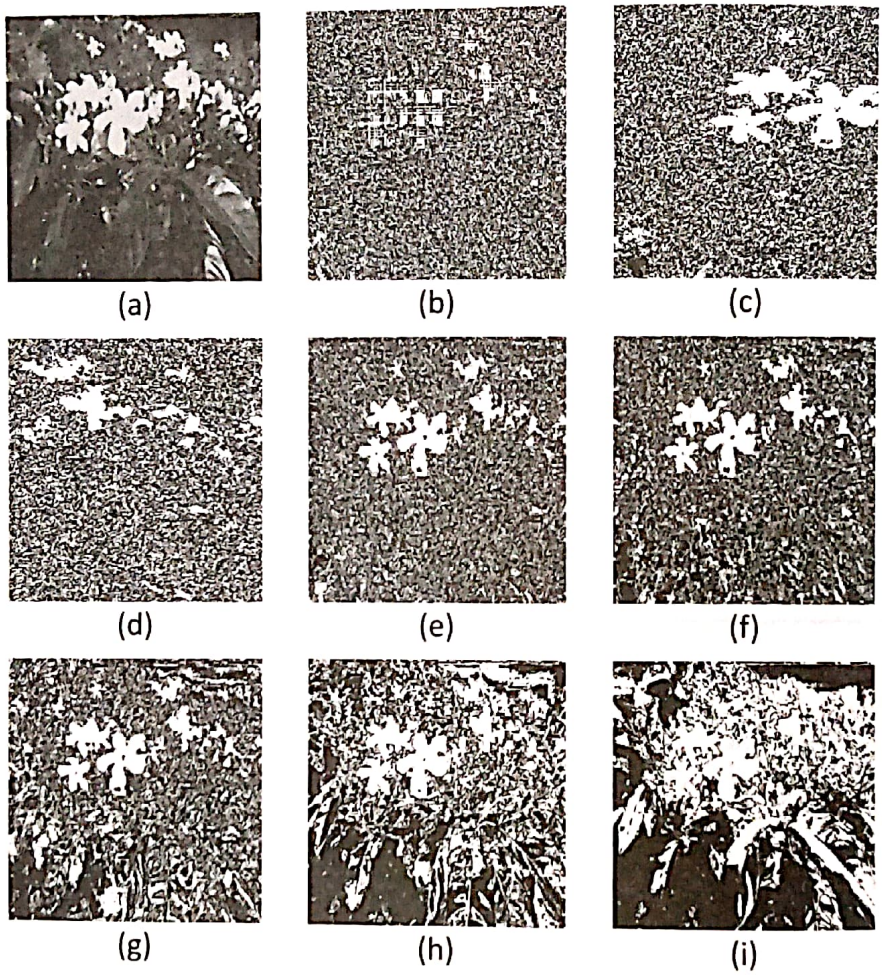


Fig. 7.15 Bit-plane coding (a) Original image (b) 0th bit plane (c) 1st bit plane (d) 2nd bit plane (e) 3rd bit plane (f) 4th bit plane (g) 5th bit plane (h) 6th bit plane (i) 7th bit plane

The individual planes of the image can now be compressed using RLC techniques. If a plane is completely white or black, RLC yields peak performance. However, the disadvantage of this scheme is that the neighbours in the spatial domain, say 3 and 4, having binary codes 011 and 100, are not present together in any of the planes. So it is reasonable to expect that the close neighbours in the spatial domain not being present together in the bit plane causes a problem. In addition, small changes in the grey level have a chain effect and the complexity of the bit plane is significantly affected.

To avoid this problem, grey code can be used instead of binary code. In grey code, the successive codes differ by only one bit. The algorithm for generating grey code can be given as follows:

$$g_i = \begin{cases} a_i \oplus a_{i+1} & 0 \le i \le m-2 \\ a_i & i = m-1 \end{cases}$$

Once the planes are obtained separately, RLC can be applied to the individual planes.

Another coding scheme that can be used for the bit plane is constant area coding (CAC). The bit planes have uniform regions of 1s and 0s. A constant portion of the bit plane can be uniquely coded using less number of bits. CAC divides the image into a set of blocks of size $(m \times m)$ like 8×8 or 16×16. There are three types of blocks available:

1. A block of all white pixels
2. A block of all black pixels
3. A block with mixed pixels

The most probable block is assigned a single code of either 0 or 1. The remaining blocks are assigned a 2-bit code. White block skipping (WBS) is a scheme where a majority of the blocks (white) are assigned a single bit. The rest of the blocks including mixed pixel blocks are encoded together.

7.4.5 Arithmetic Coding

Arithmetic coding is another popular algorithm and is widely used, like the Huffman coding technique. The differences between arithmetic and Huffman coding are shown in Table 7.16.

Table 7.16 Differences between arithmetic coding and Huffman coding

Arithmetic coding	Huffman coding
This is a complex technique for coding short messages.	This is a simple technique for coding characters.
It is always optimal.	It is optimal only if the probabilities of the symbols are negative powers of two.
Precision is a big issue.	Precision is not an important factor.
There is no slow reconstruction.	There is slow reconstruction when the number of symbols is very large and changing rapidly.

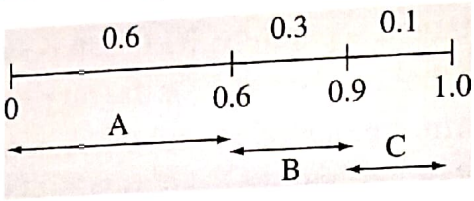
Arithmetic coding uses a single code word for a string of characters. Consider the symbols and their probabilities given in Table 7.17.

Table 7.17 Character probability table

Character	Probability
A	0.6
B	0.3
C	0.1

Let us try to code the string 'CAB'. The arithmetic coding process is carried out as follows:

1. The first step is to divide the range into 0–1 based on the probabilities.

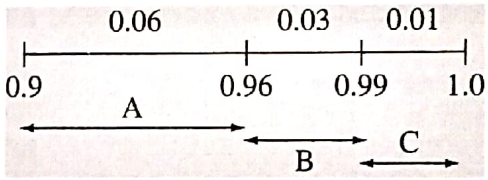


The first character to be encoded is C. This falls in the range 0.9–1.0. So the code would start in this range only. The range here is 1.0 – 0.9 = 0.1. This range of 0.1 is now divided among the symbols according the probability given. The new range can be obtained by multiplying the probability and 0.1. The cumulative probability is given as in Table 7.18.

Table 7.18 Cumulative character probability table

Character	Cumulative probability
A	$0.9 + 0.6 \times 0.1 = 0.96$
B	$0.96 + 0.3 \times 0.1 = 0.99$
C	$0.99 + 0.1 \times 0.1 = 1.0$

2. The resultant new range is 0.96–1.00. This is given as follows:

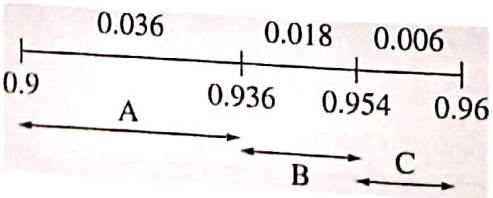


Now the second character to be encoded is A. This falls in the range 0.9–0.96. So the code would start in this range only. The range here is 0.96–0.9 = 0.06. This range is now divided among the symbols according to the probability given (Table 7.19).

Table 7.19 Cumulative character probability table

Character	Cumulative probability
A	$0.9 + 0.6 \times 0.06 = 0.936$
B	$0.936 + 0.3 \times 0.06 = 0.954$
C	$0.954 + 0.1 \times 0.1 = 0.96$

3. The resultant range is as follows:

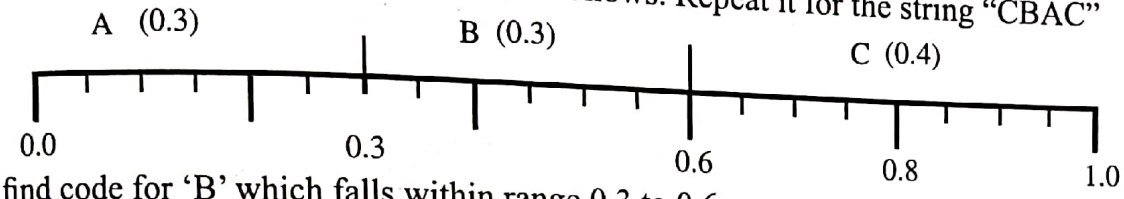


The next character to be encoded is B. This falls in the range 0.936–0.954. With this character this string ends. So the final code for the string 'CAB' is between 0.936 and 0.954. By this logic, any string can be represented.

Example 7.10 Construct Arithmetic coding for the string "BCAB' using the following table.

Symbols	A	B	C
Probability	0.3	0.3	0.4

Solution The initial mapping would be as follows. Repeat it for the string "CBAC"

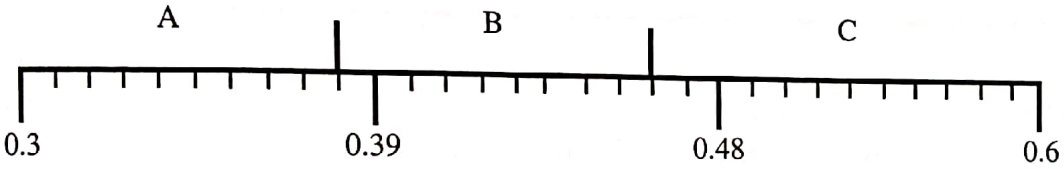


To find code for 'B' which falls within range 0.3 to 0.6

$A = 0.3 + 0.3 \times 0.3 = 0.3 + 0.09 = 0.39$

$B = 0.39 + 0.3 \times 0.3 = 0.39 + 0.09 = 0.48$

$C = 0.48 + 0.3 \times 0.4 = 0.48 + 0.12 = 0.6$

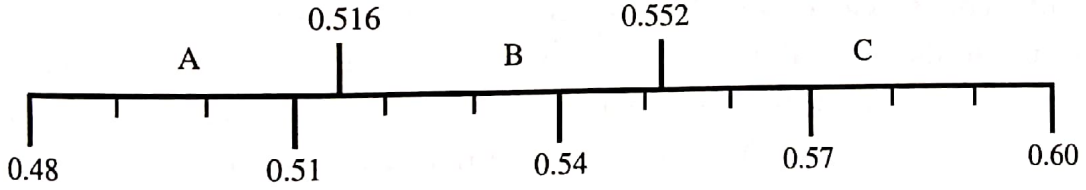


To code for 'C' which is in the range 0.48 to 0.6

$A = 0.48 + 0.3 \times 0.12 = 0.516$

$B = 0.76 + 0.3 \times 0.12 = 0.552$

$C = 0.552 + 0.4 \times 0.12 = 0.6$

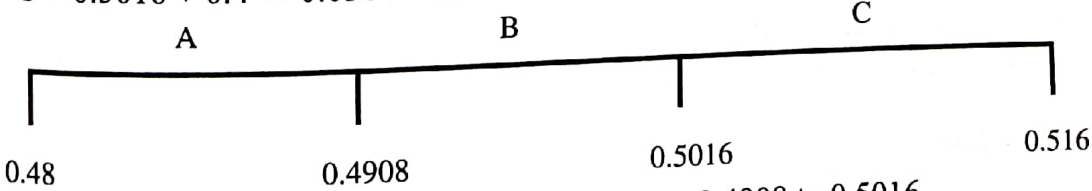


To code for 'A' in range 0.48 to 0.516

$A = 0.48 + 0.3 \times 0.036 = 0.4908$

$B = 0.4908 + 0.3 \times 0.036 = 0.5016$

$C = 0.5016 + 0.4 \times 0.036 = 0.6$



The next character to code is B which lies in the range 0.4908 to 0.5016

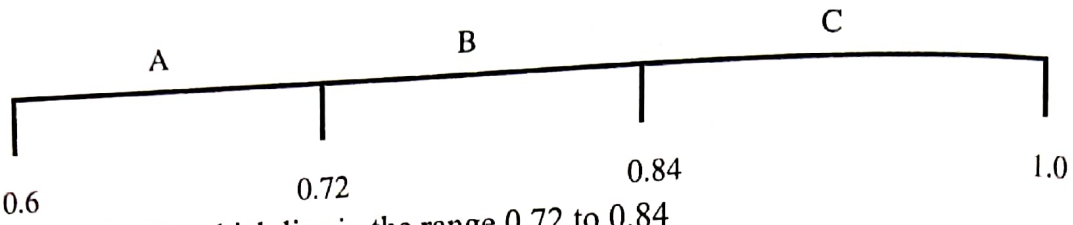
Repeat it to code for 'CBAC'

The first character is 'C' which lies in the range 0.6 to 1.0

$A = 0.6 + 0.3 \times 0.4 = 0.72$

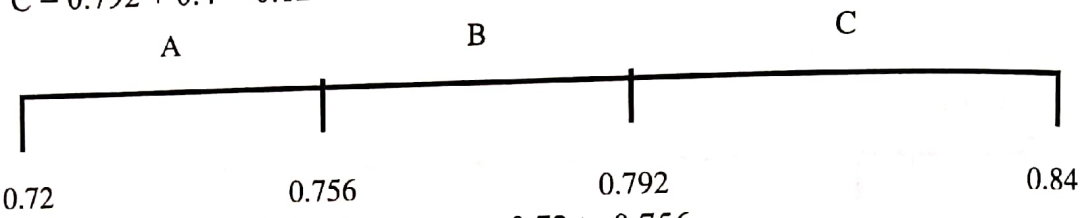
$B = 0.72 + 0.3 \times 0.4 = 0.84$

$C = 0.84 + 0.4 \times 0.4 = 1.0$



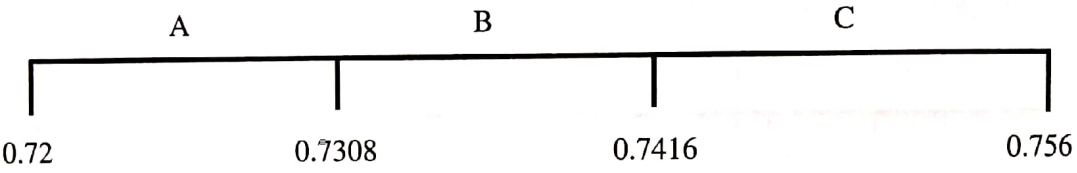
To find code for 'B' which lies in the range 0.72 to 0.84

$A = 0.72 + 0.3 \times 0.12 = 0.756$
$B = 0.756 + 0.3 \times 0.12 = 0.792$
$C = 0.792 + 0.4 \times 0.12 = 0.84$



To find code for 'A' which lies in the range 0.72 to 0.756

$A = 0.72 + 0.3 \times 0.036 = 0.7308$
$B = 0.7308 + 0.3 \times 0.036 = 0.7416$
$C = 0.7416 + 0.4 \times 0.036 = 0.756$



To find code for 'C' which lies in the range 0.7416 to 0.756

7.4.6 Dictionary-based Coding

The idea behind Lempel–Ziv–Welch (LZW) coding is to use a dictionary to store the string patterns that have already been encountered. Indices are used to encode the repeated patterns. The encoder reads the input string. Then it identifies the recurrent words and outputs their indices from the dictionary. If a new word is encountered, the word is sent as output in the uncompressed form and is entered into the dictionary as a new entry. The advantages of the dictionary-based methods are as follows:

1. They are faster.
2. These methods are not based on statistics. Thus there is no dependency of the quality of the model on the distribution of data.
3. These methods are adaptive in nature.

7.4.6.1 Encoding

The idea is to identify the longest pattern for each collected segment of the input string. It is then checked in the dictionary. If there is no match, the segment becomes a new entry in the dictionary. The algorithm is as follows:

1. Word = NULL
2. While not end of the file

Read next character and assign it to K
If (Word + K) is in the dictionary, then
    Word = Word + K
Else
    Output the dictionary index for Word
    Add (Word + K) to the dictionary
    Assign K to word
End If
End While

3. Transmit the dictionary index for the word.

Example 7.11 Consider the string ADBB and illustrate the working of the dictionary-based method.

Solution The trace of the algorithm is shown in Table 7.20.

1. Word = Null
2. Read the string.
   $K \leftarrow$ A
   Word + K = A; This is already in the dictionary.
   Therefore the word is
   Word = Word + A = Null + A = A
3. Read the string again: D B B
   Hence K = D
   Word + K = AD; This is not in the dictionary.
   Thus send output = 1
   New entry = A + D = AD; Add it to the dictionary.
   Now Word = D
4. Read the string again: B B
   Read next character K = B
   Word + K = DB; D is already in the dictionary
   Thus send output = 3
   New entry = DB; Add it to the dictionary.
   Word = B
5. Read the string again: B
   Symbol to be read K = B
   Word + K = BB; B is already in the dictionary
   Output 2
   New entry = BB; Add it to the dictionary.
   Word = B
6. Symbol to be read = Null
   Output 2
   Stop

The code that will be generated is 1 3 2 2.

Table 7.20 Steps for encoding in dictionary-based method

	Initial dictionary	Dictionary with new entry AD	Dictionary with new entry DB	Dictionary with new entry BB
A	1	1	1	1
B	2	2	2	2
D	3	3	3	3
AD	–	4	4	4
DB	–	–	5	5
BB	–	–	–	6

7.4.6.2 Decoding

The decoding algorithm for the dictionary-based method is as follows:

1. Initially read the token (or element) x and output its dictionary entry.
2. Assign element to word.
3. Read the token (or element) k.
4. While element k null repeat steps 5–8.
5. Assign the dictionary entry of the token to entry.
6. Output the value of entry.
7. Add Word + entry[0] to the dictionary.
8. Update Word as Word = element.
9. End While.
10. Exit.

The trace of the algorithm is shown in Table 7.21.

1. Start with the dictionary.
   Read token $x = 1$
   Element = A
   Output = A
   Word = A
2. Tokens to be read = 3 2 2
   Read token $x = 3$
   Element = D
   Output = D
   Add new entry to dictionary (AD = 4).
   Word = D
3. Tokens to be read = 2 2
   Read token $x = 2$
   Element = B

Output = B

Add new entry to dictionary (DB = 5).

Word = B

4. Tokens to be read = 2

   Element = B

   Output = B

   Add new entry to dictionary (BB = 6).

   Word = B

5. Tokens to be read = Null

   Stop

Table 7.21 Steps for decoding in dictionary-based method

	Initial dictionary	Dictionary with new entry AD	Dictionary with new entry DB	Dictionary with new entry BB
A	1	1	1	1
B	2	2	2	2
D	3	3	3	3
AD	–	4	4	4
DB	–	–	5	5
BB	–	–	–	6

Thus the algorithm retrieves the original and constructs the dictionary adaptively.

7.4.7 Lossless Predictive Coding

Predictive coding techniques eliminate the inter-pixel dependencies by predicting new information, which is obtained by taking the difference between the actual and the predicted value of that pixel. The encoder takes a pixel of the input image f_n. The predictor predicts the anticipated value of that pixel using past inputs (historical data). The predicted value is rounded to the nearest integer value denoted by $\hat{f}_n$. This is the predicted value. The error is the difference between the actual and the predicted values.

$$e_n = f_n - \hat{f}_n$$

This error is sent across the channel. The same predictor is used in the decoder side to predict the value. The reconstructed image is

$$f_n = e_n + \hat{f}_n$$

Example 7.12 Apply differential encoding for the following sequence

6.2 9.7 13.2 5.9 8.0 7.4 4.2 1.8

Solution The first pixel would be transmitted as it is and then the differences would be transmitted. The following steps are done. The transmitted values would be

$$\begin{aligned} & 6.2 \\ (9.7-6.2) &= 3.5 \\ (13.2-9.7) &= 3.5 \\ (5.9-13.2) &= -7.3 \\ (8.0-5.9) &= 2.1 \\ (7.4-8.0) &= -0.6 \\ (4.2-7.4) &= -3.2 \\ (1.8-4.2) &= -2.4 \end{aligned}$$

After quantization:
In lossy scheme, this is rounded as

$$\{6, 4, 4, -7, 2, 0, -3, -2\}$$

The reconstructed signal would be

$$6, 10, 14, 7, 9, 9, 6, 4$$

This results in error of

$$0.8, -0.3, -0.8, 1.1, -1, -1.6, -1.8, -2.2$$

The most crucial part of the design is that of the predictor. The prediction by a linear predictor taking a linear estimation of the previous n bits is given as

$$\hat{f}_n = \text{round}\left\{\sum_{i=1}^{m} \alpha_i f_{n-i}\right\}$$

Here m is the order of the predictor as a function. The 1D linear predictive coding $\hat{f}_n$ can be written as

$$\hat{f}_n(x,y) = \text{round}\left[\sum_{i=1}^{m} \alpha_i f(x, y-i)\right]$$

The presence of a quantizer takes the prediction error into a limited range of the output denoted as e_n, which establishes the amount of computation and distortion. The predictions at the source and the destination are equal. The corresponding quantized error is

$$\dot{f}_n = e_n + \hat{f}_n$$

Example 7.13 Consider the pixels {23, 34, 39, 47, 55, 63} and demonstrate the predictive coding algorithms shown in Table 7.22.

Table 7.22 Symbols and their predictive coding difference

Value	Predictive coding
23	23
34	34 – 23 = 11
39	39 – 34 = 5
47	47 – 39 = 8
55	55 – 47 = 8
63	63 – 55 = 8

Solution The maximum value in the original sequence, that is, 63, requires 6 bits and the sign requires 1 bit. Thus the number of bits required to code the original sequence is $6 \times 7 = 42$. However, in the predictive coded sequence, the maximum value is 23, which requires only 5 bits. Considering the sign bit also, the numbers of bits required is $6 \times 6 = 36$. Hence this saves 6 bits. If the sequence becomes increasingly large, this method of coding the difference proves to be a great advantage.

The number of bits used for coding the pixel difference is 6 bits (1 for the sign bit and 5 for the pixel value). If the difference crosses the threshold limit, this proves to be a problem and is called overloading. One possible solution to overcome this is to ignore the differences and use the original message for coding. Suppose that Example 7.13 is changed as shown in Table 7.23.

Table 7.23 Predictive coding

Value	Predictive coding
23	23
64	64 – 23 = 41 (crosses the threshold of 5 bits, hence original value is used)
39	39 – 64 = –25
47	47 – 39 = 8
55	55 – 47 = 8
63	63 – 55 = 8

The problem of overloading is solved by allotting the required 7 bits to code the original value 64. So the total number of bits used is $5 \times 6 + 1 \times 7 = 37$. Thus the percentage of savings is reduced.

7.5 LOSSY COMPRESSION ALGORITHMS

Lossy compressions algorithms, unlike lossless compression algorithms, incur a loss of information. This loss is called distortion. However, this data loss is acceptable if it is

tolerable. The compression ratio of these algorithms is very large. Some popular lossy compression algorithms are as follows:

1. Lossy predictive coding
2. Vector quantization
3. Block transform coding

Lossy predictive coding is an extension of the idea of predictive coding discussed in Section 7.4.7.

7.5.1. Lossy Predictive Coding

Predictive coding can also be implemented as a lossy compression scheme. Instead of taking precautions, the highest value for 5 bits, that is, 31 can be used. This drastically reduces the number of bits. However, loss of information increases. This is illustrated in Table 7.24.

Table 7.24 Predictive coding with overloading

Value	Lossy predictive coding
23	23
64	64 – 23 = 41 (crosses the threshold of 5 bits). However, store only 31 supported by 5 bits + one sign bit = 6 bits.
39	39 – 64 = –25
47	47 – 39 = 8
55	55 – 47 = 8
63	63 – 55 = 8

Here the number of bits used to transmit is same as the original scheme, but the value 31 is transmitted instead of 41, leading to error. This loss of information leads to an error which results in a lossy compression scheme. This scheme requires only $6 \times 6 = 36$ bits.

Delta modulation Delta modulation goes one step further by using only one bit for representing the quantized error value. This can be positive or negative. Here the predictor is defined as

$$\hat{f} = \alpha \times f_{n-1}$$

where α is called the prediction coefficient. More generally, for the first digit,

$$\hat{f} = \hat{f}_{n-1}$$

Then the error is computed as follows:

$$e_n = f_n - \hat{f}_n = f_n - \hat{f}_{n-1}$$

The error is quantized as follows:

$$e_n = \begin{cases} +\zeta \text{ for } e_n > 0 \\ -\zeta \text{ otherwise} \end{cases}$$

Here ζ is the positive quantity. This scheme creates problems if significant transitions in data are encountered frequently. When ζ is varied, it is called adaptive delta modulation scheme.

Example 7.14 Apply Delta modulation scheme for the following sequence of elements

$$f = \{10,14,15\}, \text{ let } \zeta = 3$$

Solution $f_1 = 10, f_2 = 14$, and $f_3 = 15$ Initially, the first pixel would be transmitted as it is. Therefore,

$$\hat{f}_1 = f_1 = 10$$

Then, $\hat{f}_2 = 10$, Therefore, $e_2 = 14 - 10 = 4$ (as $f_2 = 14$). As this is greater than 0, $\bar{e}_2 = 3$. This would be reconstructed as $\tilde{f}_2 = 10 + 3 = 13$.

Then, $\hat{f}_3 = 13$, therefore, $e_3 = 15 - 13 = 2$ (as $f_3 = 15$), As this is greater than 0, $\bar{e}_3 = 3$. This would be reconstructed as. $\tilde{f}_3 = 13 + 3 = 16$

The reconstructed sequence would be {10.13.16}.

7.5.2 Vector Quantization

Vector quantization (VQ) is a technique similar to scalar quantization. In scalar quantization, the individual pixels are quantized. The idea of VQ is to identify the frequently occurring blocks in an image and to represent them as representative vectors. The set of all representative vectors is called the code book, which is then used for image. The structure of VQ is shown in Fig. 7.16.

The codebook formation procedure is as follows:

1. Vector quantizaion first partitions the input space X into K non-overlapping regions. It then assigns a code vector for each cluster. The code vector is commonly chosen as the centroid of the vectors of the partition.

$$C = (c_1, c_2, \ldots, c_m) = (\overline{X_1}, \overline{X_2}, \ldots, \overline{X_m})$$

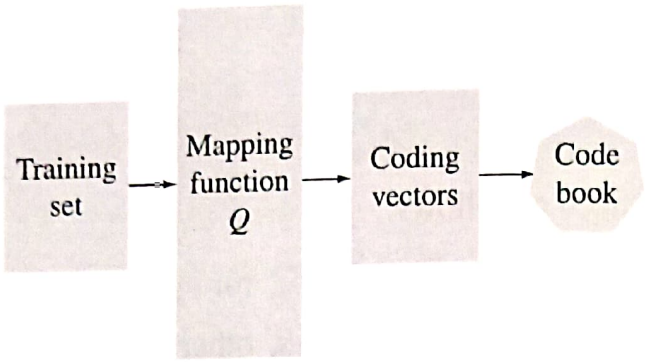


Fig. 7.16 Basic structure of vector quantization

2. It carries out a mapping process between the input vectors and the centroid vector.
3. This introduces an error called distortion measure. This distortion is described as

$$d(X,Y)=\sqrt{\sum_{i=1}^{M}(X_i-Y_i)^2}$$

Here X and Y are two M-dimensional vectors.

4. The codebook of vector quantization consists of all the code words. The image is then divided into fixed size blocks (vectors), typically 4×4, and replaced with the best match found in the codebook based on the minimum distortion.

For example, in the case of 4×4 pixel blocks and a codebook of size 256, the bit rate is $(\log_2 256)/16 = 0.5$ bpp and the corresponding compression ratio is $(16 \times 8)/8 = 16$.

7.5.2.1 Codebook design

The basis for codebook construction is the training set of the vectors. Sample vectors from different images are collected and these form the training set. The code book is constructed from these with the constraint that the average distortion is minimized with respect to the training set. One simple design is to randomly select K vectors from the training set. Though it is not efficient, it can serve as the initialization for many vector quantization algorithms.

7.5.2.2 Generalized Lloyd algorithm

This algorithm is also known as *LBG* (Linde–Buzo–Gray) *algorithm* and is described as follows:

1. Take a codebook as input.
2. Find the nearest locally optimal codebook with respect to the initial one. The initial codebook can be generated randomly.
3. Apply the nearest neighbour condition so that the training vector is mapped to its nearest code vector with respect to the distortion function—the training set is divided into partitions by mapping each vector $\boldsymbol{X}$ to its nearest code vector $\boldsymbol{Y}$ using the Euclidean distance. Alternatively, use the centroid condition that the optimal code vector is the centroid of the vectors within the partition—the centroid of each region is calculated and replaced by the centroids of their corresponding partitions.

Thus at every iteration the codebooks become progressively better. This process is continued till there is no change in the overall distortions. Even though this algorithm does not guarantee the global optimum, it can be seen that the output soon converges into a local minimum.

7.5.3 Block Transform Coding

Block transform coding is another popular lossy compression scheme. The process of transform coding is shown in Fig. 7.17.

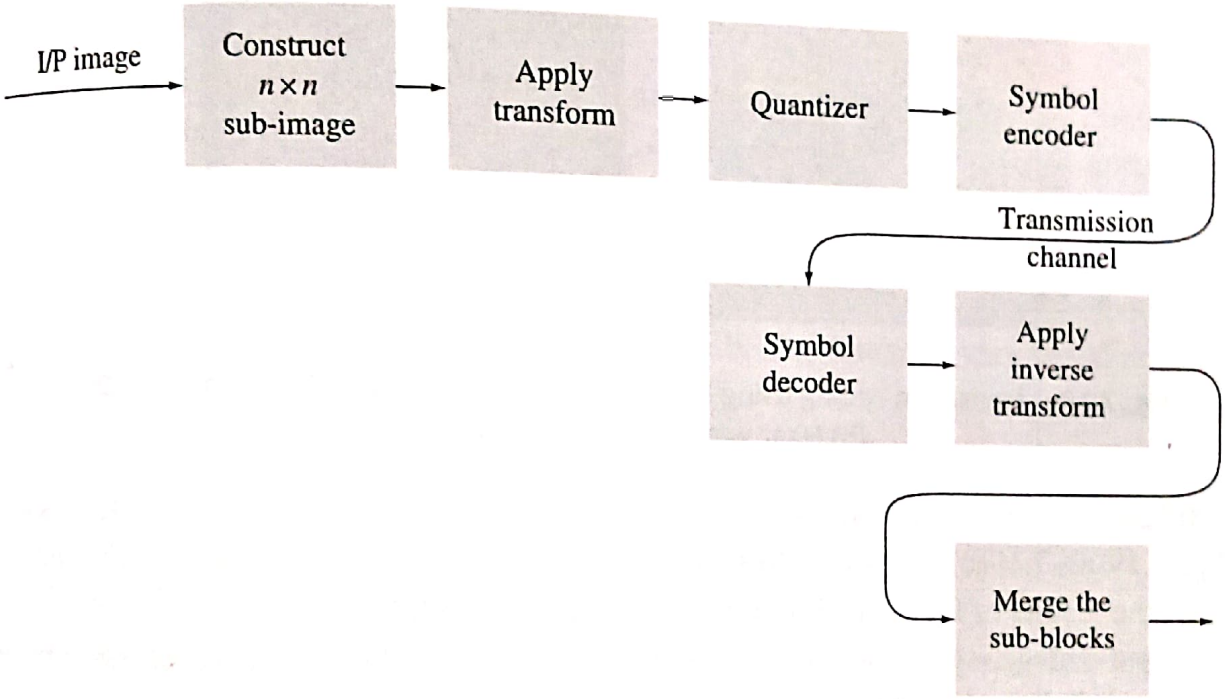


Fig. 7.17 Transform coding model

7.5.3.1 *Sub-image selection*

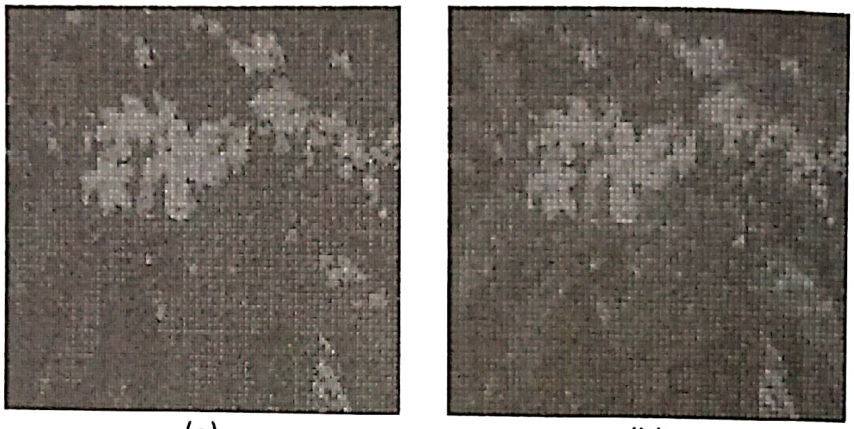
The aim of this step is to reduce the correlation between adjacent pixels to an acceptable level. This is one of the most important stages where the image is divided into a set of sub-images. As the first step, the $N \times N$ image is decomposed to a set of sub-images of size $n \times n$ for operational convenience. The value of n is a power of two. This is to ensure that the correlation among the pixels is minimum. This step is necessary to reduce the transform coding error and computational complexity. (Imagine how difficult it would be to handle matrices of size 1024×1024!) Generally sub-images would be of size 8×8 or 16×16.

7.5.3.2 *Transform selection*

The whole idea of transform coding is to use mathematical transforms for data compression. Transformations such as discrete Fourier transform (DFT), discrete cosine transform (DCT), and wavelet transforms can be used. The choice of the transforms depends on the resources and the amount of error associated with the reconstruction process. Mathematical transforms are tools for information packing.

Block transform coding using Haar wavelet transform is illustrated in Figs 7.18(a) and 7.18(b).

An important aspect of the image is that smoother details are low frequency components. Sharp details like edges are high frequency components. Transforms convert data into frequency components. Then the required frequency components are selected so that the coefficients associated with the details that are insensitive to the human eye are discarded.



(a) (b)

Fig. 7.18 Transform coding using Haar wavelet (a) Haar wavelet—block 8 × 8 (b) Haar wavelet—block 16 × 16

It has been observed that DCT offers better information packing capacity, as shown in Figs 7.19(a)–7.19(c). Figure 7.19(a) shows the original image and Figs 7.19(b) and 7.19(c) show the results of DCT in a block transform coding. KL transforms are also effective, but the disadvantage is that they are data-dependent. The digital cosine transform is preferred because it is faster and hence can pack more information.

(a) (b) (c)

Fig. 7.19 Transform coding (a) Original image (b) DCT block 8 × 8 (c) DCT block 16 × 16

7.5.3.3 Bit allocation

Transform coding is the process of truncating, quantizing, and coding the coefficients of the transformed sub-image. It is necessary to assign bits such that the compressed image will have minimum distortions. Bit allocation should be done based on the importance of

the data. The idea of bit allocation is to reduce the distortion by optimal allocation of bits to the different classes of data. The steps involved in bit allocation are as follows:

1. Assign predefined bits to all classes of data in the image.
2. Reduce the number of bits by one and calculate the distortion.
3. Identify the data that is associated with the minimum distortion and reduce one bit from its quota.
4. Find the distortion rate again.
5. Compare with the target and if necessary repeat steps 1–4 to get the optimal rate.

7.5.3.4 Zonal coding

The zonal coding process involves multiplying each transform coefficient by the corresponding element in a zonal mask, which has 1 in the location of maximum variance and 0 in the other places. A zonal mask is designed as part of this process. These coefficients are retained as they convey more image information. The locations are identified based on the image models used for source symbol encoding. The retained coefficients are quantized and coded. The number of bits allocated may be fixed or may vary based on some optimal quantizer.

$$M(u,v) = \begin{cases} 1 \text{ if } T(u,v) \text{ has the largest coefficient} \\ 0 \text{ otherwise} \end{cases}$$

7.5.3.5 Threshold mask

Threshold coding works based on the fact that transform coefficients having the maximum magnitude make the most contribution to the image. The threshold may be one of the following:

1. A single global threshold
2. An adaptive threshold for each sub-image
3. A variable threshold as a function of the location for each coefficient in the sub-image

The thresholding and quantization processes can be combined; their approximation is

$$\hat{T} = \text{Round}\left(\frac{T(u,v)}{Z(u,v)}\right)$$

$Z(u, v)$ is the transform normalized array.

$$\dot{T} = \hat{T}(u,v) \times z(u,v)$$

The inverse transform of $\dot{T}$ gives the decompressed image approximately. It is assumed that the largest magnitude makes the most significant contribution. This varies from one mask to another. The mask has 1 in the place of the maximum threshold and 0 in other places.

Example 7.15 Assume a quantization threshold of 32 and derive the quantization error for each of the following DCT coefficients:

127 72 67 78 128 168

Solution The thresholding table is given in Table 7.25. The quantization value can be obtained using the formula [DCT coefficient/quantization threshold] and then rounding it to the nearest integer. The dequantization value can be obtained by multiplying the rounded off value and the threshold coefficient. The difference between the original value and the quantized value is called error.

Table 7.25 Thresholding table

Value	Quantization	Dequantized value	Error
127	$\frac{127}{32}=3.96875 \cong 4$	$4 \times 32 = 128$	−1
72	$\frac{72}{32} = 2.25 \cong 2$	$2 \times 32 = 64$	8
67	$\frac{67}{32} = 2.09 \cong 2$	$2 \times 32 = 64$	3
128	$\frac{128}{32} = 4$	$4 \times 32 = 128$	0
168	$\frac{168}{32} = 5.25 \cong 5$	$5 \times 32 = 160$	8

The error would always range between ±50% of the threshold value.

7.6 IMAGE AND VIDEO COMPRESSION STANDARDS

Similar to fax compression standards discussed in Section 7.4, knowledge of image compression standards for images and video is important in image processing. Joint photographic experts group (JPEG) is a popular image compression standard. JPEG is a family of compression algorithms. The International Standards Organization (ISO) and the International Telecommunication Union (ITU) jointly proposed this standard, which became an International standard in 1992.

Moving pictures experts group (MPEG) is a video compression standard proposed by the ISO. Video has both visual and audio content. The MPEG standard deals with compression, decompression, and processing of both audio and video streams.

7.6.1 JPEG

The JPEG format offers the following four modes of operation:

1. Sequential DCT-based mode (baseline algorithm)
2. Lossless mode
3. Progressive DCT-based mode
4. Hierarchical mode

7.6.1.1 Sequential DCT-based mode (baseline algorithm)

The baseline algorithm consists of the following steps:

1. Image preparation
2. Application of image transforms
3. Quantization
4. Entropy encoding
5. Frame building

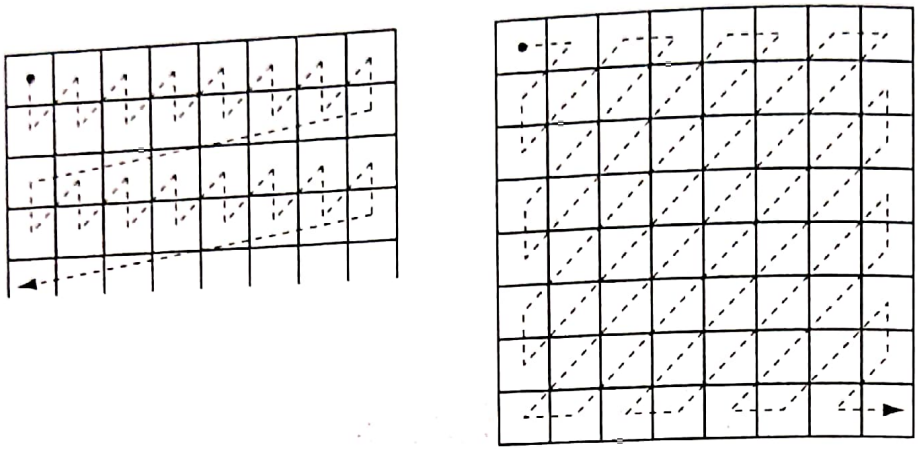
Image preparation The input image may be a monochrome or a true colour image. The colour components can be subsampled. Hence it is necessary to convert an RGB image into a YCbCr (luminance–chrominance–colour space) image, so that the colour components are separated from the luminance part. Similar to transform coding, images are divided into smaller blocks. For example, if the RGB image is 1024×1024, it can be divided into subblocks of 4×4, 8×8, or 16×16. If the sub-block size chosen is 8×8, there will be $1024/8 \times 1024/8 = 128 \times 128$ blocks in the horizontal and vertical directions.

Image transforms The inputs for the transforms are the blocks from the image preparation step. The output of the transforms has the same size as the original image. Image transforms like DCT or wavelet can also be used.

Quantization After the frequency coefficients are obtained, all the frequency components are not necessary since the human eye is sensitive only to the low frequency components. For this purpose, a threshold value is applied. The frequency components whose values are lesser than the threshold value are discarded. The quantization table is used for this purpose. If the value in the quantization table is 10 and the frequency coefficient is 43, the quantized table value is $43/10 = 4.3$. This is approximated to the nearest integer, that is, 4. Depending on the threshold value, it is either retained or discarded.

Entropy encoding The quantized values are encoded using entropy coding techniques. If an 8×8 block is assumed, then the first element (0, 0) is called the DC coefficient and the other 63 elements are called AC coefficients. The DC coefficient has larger amplitude and captures the average information of the image. Differential encoding is used for encoding the DC coefficient. The remaining 63 AC coefficients are fast-varying data. These are arranged in a zigzag sequence. Some sample zigzag patterns are shown in Fig. 7.20. First, run-length coding is applied to the sequence. Then the output of the RLC algorithm is Huffman-coded for more compression.

Frame building This is the final stage, where the transmission requirements are taken into account. Transmission requires adherence to certain rules and regulations. Therefore, the frame header is created with additional information such as start bit, end bit, data type, and nature of the image. The compressed data is packed and the frames are sent across the channel.



(a) (b)

Fig. 7.20 JPEG entropy encoding—sample zigzag masks

The JPEG decoder is the same as the encoder. It consists of the following components:

1. Frame decoder
2. Entropy decoder
3. Dequantization
4. Inverse DCT transform
5. Image builder

These components, along with the quantization table, form the decoder part where every component does the reverse of the corresponding encoder components. Due to quantization, information loss occurs, and hence perfect reconstruction is not possible. However, the thresholds are such that the difference is not noticeable by the human visual system.

7.6.1.2 Lossless mode

JPEG also offers lossless compression. This mode creates a perfect duplicate of the original image. This is shown in Fig. 7.21, where X is the pixel to be predicted using the pixels Q, S, and T.

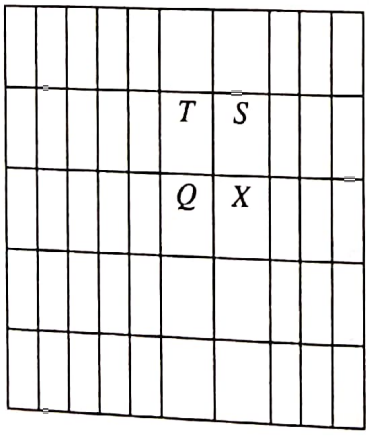


Fig. 7.21 JPEG lossless mode

The ways of predicting the unknown pixel X using the neighbour pixels Q, S, and T is shown in Table 7.26.

Table 7.26 JPEG lossless scheme

Prediction scheme	Prediction of the pixel X (P_x)
1	No prediction
2	Q
3	S
4	T
5	$Q+S-T$
6	$Q+\frac{(S-T)}{2}$
7	$S+\frac{(Q-T)}{2}$
8	$\frac{(Q+S)}{2}$

7.6.1.3 Progressive encoding

The main idea of progressive encoding is a gradual compression using the priority of the pixels. First it sends the coarse version of the image to the receiver. Then the additional information is sent so that the quality of the image is progressively refined. The advantages of this scheme are that the user can control the amount of loss or image quality by controlling the stop process of the encoder. This is useful in internet applications where the user is not prepared for wait a longer time to download an image. So it is better to get the coarse version of the image and as the user is prepared to wait longer, information of better quality can be sent. This mode is similar to the sequential DCT mode in all aspects. The subblocks are DCT coded and then the DCT coefficients are stored in a buffer. Then the frequency coefficients are divided into multiple spectral bands. The spectral bands are decided based on the zigzag scanning order. Then this mode uses a progressive scheme known as spectral selection where low frequency coefficients are sent first and then the remaining coefficients. Another scheme is known as successive scheme where the most significant bits of the quantized coefficients are sent first. Then the additional less significant bits of the pixels are sent.

7.6.1.4 Hierarchical mode

This scheme uses the pyramidal data structure that stores the image at several different resolutions. Pyramidal data structures are discussed in Chapter 3 whose advantage is that they allow the user to negotiate with the application at the required resolution. Hence this mode is useful for imaging applications where there is a need for multiresolution requirements. The bottom layer is the original image in a pyramid data structure and the subsequent layers are a subsampled image of the original, as every layer differs from the subsequent layer by a sampling factor of 2. Thus predictive coding is used to encode the differential frames. Hence hierarchical coding supports lossless code as well as progressive coding.

SUMMARY

- Images and video objects take a lot of space and transmission of images and videos takes time. Hence image compression is necessary for image storage and transmission.
- Data and information are two different things. Data is raw fact. Information is processed data.
- Redundant data is repetitive and does not convey much information.
- Information theory uses probability to investigate information and aims to measure it.
- Lossless compression is useful in preserving information as there is no information loss. This type of algorithm is useful in legal and medical domains. Lossy compression algorithms compress the data with a certain amount of error that is acceptable to the human observer.
- The logic behind entropy encoding is that if pixels are not uniformly distributed, a code can be selected that encodes the information so that it is less than the entropy. The idea behind predictive coding is to remove mutual dependency between successive pixels. The idea behind transform coding is to exploit the information packing capability of the transform.
- RLC exploits the repetitive nature of the image. It tries to identify the run of the pixel values and encodes the image in the form of a run.
- Huffman coding is a variable length coding technique. In Huffman coding, coding redundancy can be eliminated by choosing a better way of assigning codes.
- Arithmetic coding uses a single code word for a string of characters.
- The idea of LZW coding is to use a dictionary to store the string p1atterns that have already been encountered. Indices are used to encode the repeated patterns.
- Predictive coding techniques eliminate the inter-pixel dependencies by predicting new information, which is obtained by taking the difference between the actual and the predicted values of that pixel.
- Transform coding is the process of truncating, quantizing, and coding the coefficients of the transformed sub image.
- The zonal coding process involves multiplying each transform coefficient by the corresponding element in a zonal mask, which has the value 1 in the location of maximum variance and the value 0 in other places.
- JPEG is the abbreviation for Joint Photographic Experts Group and is a family of compression techniques proposed by the joint efforts of ISO and CCITT. JPEG offers four modes of operation—sequential, lossless, progressive, and hierarchical.
- Video compression is based on two important aspects—spatial and temporal redundancies.

KEY TERMS

Compression It is the process of removing coding, inter-pixel, psychovisual, and chromatic redundancies present in an image.

Compression ratio It is the ratio of the original message (or file size) and the compressed code (or file size).

Entropy coding It is the type of coding that is based on the entropy of the source.

Information content It is the quality of information present in a data source.

JPEG It is the abbreviation for joint photographic experts group and is a process used for compressing an image.

Lossless compression It refers to algorithms that preserve data with no loss of information.

Lossy compression It refers to algorithms that incur data loss, that is acceptable to the human observer.

MPEG It is the abbreviation for motion photographic experts group and is a standard for compressing video information.

Predictive coding It is the type of coding that is based on the prediction of subsequent data.

Progressive encoding It is the process of gradual compression following an underlined order of priority.

Psychoacoustic model It is the mathematical model of the human auditory system.

Redundancy It refers to repetitive data in the data source.

Savings percentage It is the amount of savings in storage expressed in terms of percentage.

Transform coding It is the type of coding scheme that uses mathematical transforms.

Vector quantization It is the coding scheme based on blocks of data.

REVIEW QUESTIONS

1. Find the odd item out of the following data list.
   (a) Data in database
   (b) Executable file data
   (c) Image data
   (d) Video data
   (e) Text files data
   Justify.
2. What is meant by distortion? How will you characterize it?
3. Distinguish between the terms logical compression and physical compression.
4. What is the entropy of tossing a coin and rolling a dice?
5. Does the Huffman scheme always give the same code? What is the meant by prefix property?
6. How do pattern substitution methods help compression?
7. What is the need for vector quantization?
8. What is the theoretical justification of transform coding? Why does it work well?
9. Distinguish between zonal and threshold masks.

NUMERICAL PROBLEMS

1. Calculate the entropy for the following image:

Symbols	1	2	3	4	5	6
Probability	0.4	0.2	0.2	0.1	0.08	0.02

2. Construct the Huffman code for the given image data.

Symbols	1	2	3	4	5	6
Probability	0.4	0.2	0.2	0.1	0.08	0.02

3. Construct the RLC for the following image:

$$\begin{pmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 1 & 2 & 2 \end{pmatrix}$$

4. Construct the Shannon–Fano tree for the given image.

$$\begin{pmatrix} 1 & 1 & 1 \\ 2 & 1 & 1 \\ 1 & 2 & 2 \end{pmatrix}$$

5. Demonstrate the predictive coding algorithm for the following pixels:

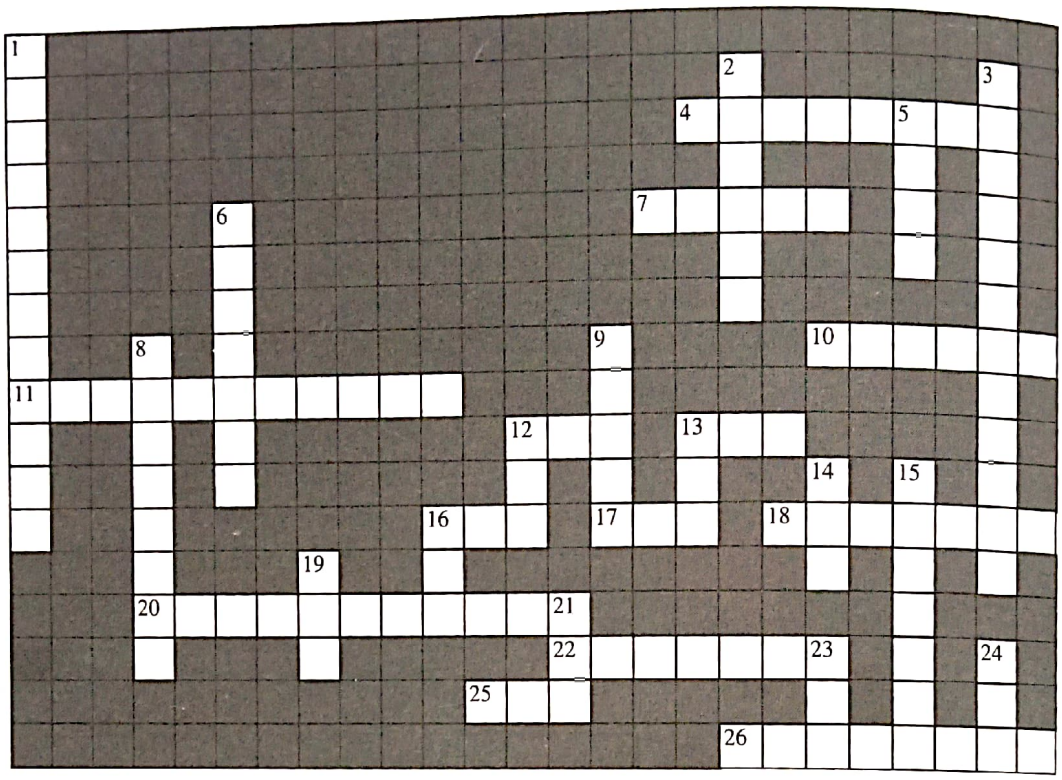
23	34	39	47	55	63

6. With the following symbols and their probabilities of occurrence, encode the message ABCD using the arithmetic coding algorithm.

Symbol	A	B	C	D
Probability	0.4	0.3	0.2	0.1

7. Code the following message using arithmetic coding algorithm:
   SWISS
8. Use LZW and encode the following message:
   ABBABAS
9. Assuming the quantization thresholds of 32, 48, and 64. Derive the quantization error for each of the following DCT coefficients:
   127, 172, 167, 178, −164, and −128

CROSSWORD



Across

4. Huffman coding is _____ length coding.
7. Compression ratio is the ratio of file size before compression and file size _____ compression.
10. The idea of predictive coding is to remove the _____ dependency among the pixels.
11. Transform coding takes the advantage of _____ packing capability of transforms.
12. Lossless predictive coding removes the inter-pixel redundancies. (Yes/No)
13. RGB is converted to YCbCr for reducing the colours. (Yes/No)
16. Two-dimensional RLC is useful for facsimile compression. (Yes/No)
17. JPEG stands for Joint Photographic Experts Group. (Yes/No)
18. Two main components of compression system are encoder and _____.
20. Chromatic redundancy refers to presence of _____ (necessary/unnecessary) colours in the image.
22. Coding redundancy is the difference between average bits to code and _____.
25. Arithmetic coding is always optimal. (Yes/No)
26. Broadly compression schemes can be classified as lossy and _____.

Down

1. Pyramidal data structures are used in _____ mode of JPEG.
2. Image compression is required for three reasons such as reduced storage, reduced bandwidth, and _____ computations.
3. Image compression is the art and science of removing _____ present in image data.
5. The unit of information content is _____.
6. Entropy represents the _____ number of bits used to encode information.
8. Shannon-Fano coding is a _____ (top-down/bottom-up) way of building the tree.
9. Vector quantization is a _____ compression scheme.
12. Precision is a big issue in arithmetic coding. (Yes/No)
13. JPEG was proposed by ISO and ITU. (Yes/No)
14. Bit rate refers to the rate of bit transfer. (Yes/No)
15. UK is used to represent United Kingdom. This kind of abbreviations is called _____ compression.
16. Run length coding exploits the repetitive nature of pixel contents in images. (Yes/No)
19. JPEG uses _____ (DCT/wavelet) transform.
21. Spatial redundancy is refered to as geometrical redundancy. (Yes/No)
23. When the probability content is 1, the information content would be zero. (Yes/No)
24. Redundancy means repetitive data. (Yes/No)

WORD SEARCH PUZZLE

Some of the important terms in this chapter are present in the following word jumble. Identify the words. Diagonal words are possible.

I	F	L	H	Q	U	C	L	U	U	N	N	E	C	E	S	S	A	R	Y	Z	N	Q	G	T
E	G	Y	R	O	Z	X	I	V	W	S	K	Y	N	Y	E	S	N	P	Y	F	Y	E	S	D
F	T	I	O	O	H	S	G	O	E	I	A	E	S	L	E	N	T	R	O	P	Y	N	A	Z
Q	O	Q	X	H	S	X	X	Y	U	D	C	S	B	A	N	C	W	P	T	M	P	4	R	C
B	U	V	F	E	L	O	S	E	L	E	S	S	X	F	R	A	S	K	B	U	P	A	E	S
S	B	I	Y	U	T	H	I	H	A	N	M	N	Q	T	S	D	H	D	Q	T	T	F	D	G
B	L	V	U	K	E	M	N	S	C	H	S	P	Y	E	H	D	Q	C	Y	U	L	V	U	Y
O	O	A	M	A	W	G	F	J	Z	J	Y	X	Y	R	N	T	Q	T	A	A	Z	E	N	E
T	G	R	K	E	H	J	O	I	O	U	A	C	X	D	W	G	Y	J	G	L	H	E	D	S
T	I	I	I	U	A	M	R	G	M	Z	S	W	C	T	J	C	E	G	Z	N	Y	S	A	D
O	C	A	F	E	M	Y	M	H	I	E	R	A	R	C	H	I	C	A	L	B	T	H	N	C
M	A	B	W	N	I	H	A	M	P	S	Y	P	K	D	E	C	O	D	E	R	O	Y	C	G
U	L	L	W	N	N	G	T	O	T	Z	E	V	D	M	Q	P	U	L	F	H	S	Y	I	U
P	E	E	X	Z	I	L	I	S	Q	D	S	C	N	B	L	X	J	S	G	E	B	E	E	B
L	O	S	S	Y	M	K	O	O	R	Z	E	Y	T	A	J	S	Y	A	Y	S	I	Q	S	C
P	E	K	D	I	U	V	N	V	U	X	Y	E	S	O	I	J	B	R	E	D	T	B	E	C
Y	F	Y	P	Y	M	R	J	D	F	A	S	T	E	R	G	B	X	Y	E	X	S	J	R	H

Hints

1. Spatial _____ removal results in compression of the image.
2. _____ component of the compression system restores the original image.
3. Abbreviations are good examples of _____ compression.
4. Absence of quantization process often results in _____ compression.
5. Entropy specifies _____ number of bits to represent an image.
6. _____ information is a metric.
7. Compressions reduce bits and not _____.
8. _____ per pixel is an indicator of compression.
9. _____ indicates the orderliness of a system.
10. Huffman code builds a tree in _____ manner.
11. _____ compression schemes result in more compression ratio.
12. JPEG uses _____ as a primary transform.
13. JPEG modes are sequential, progressive, and _____.