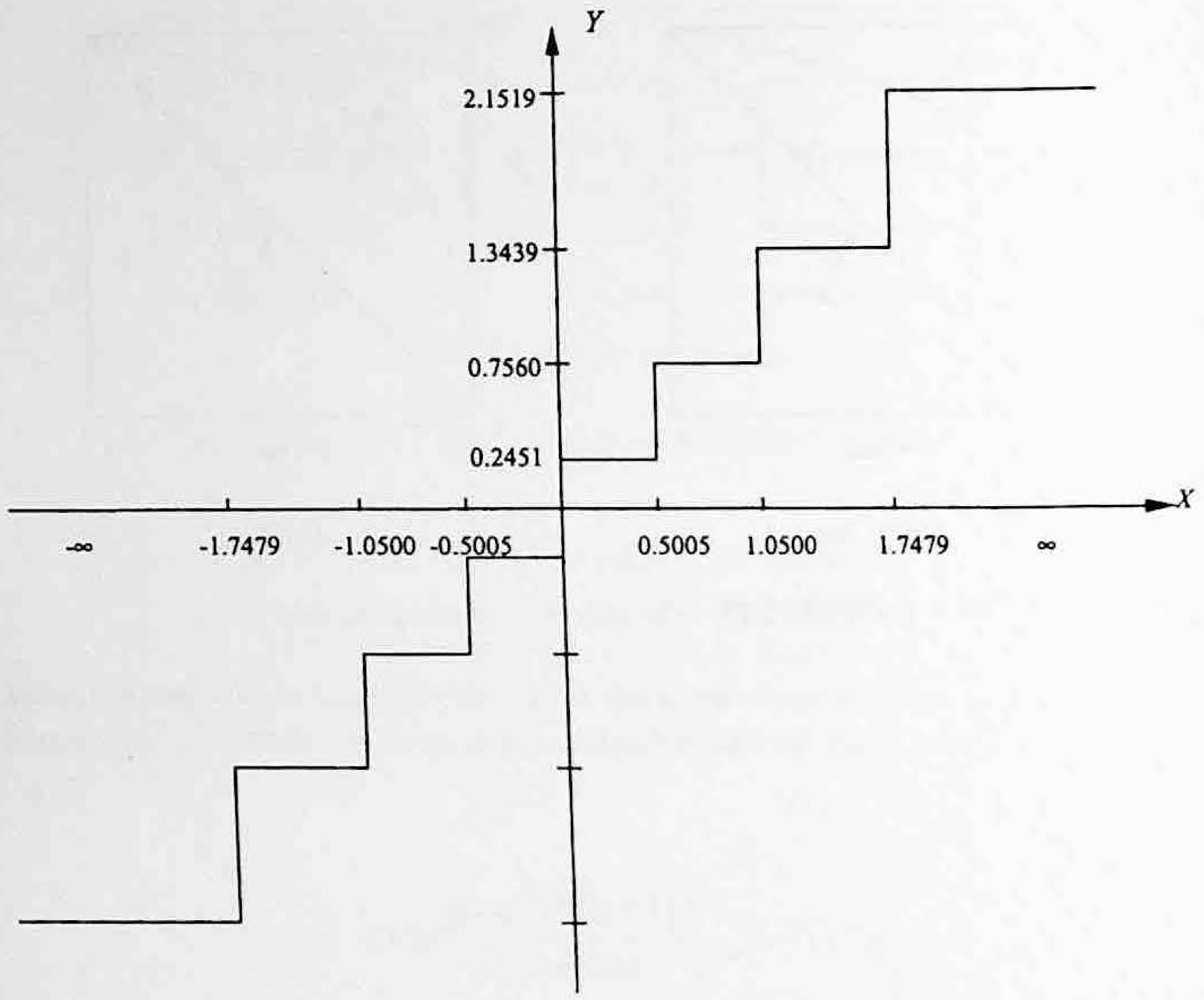


FIGURE 2.13 Input-output characteristic of the optimal quantizer for Gaussian distribution with zero mean, unit variance, and N = 8.

decision levels are densely distributed in the region having a higher probability of occurrence and coarsely distributed in other regions. A logarithmic companding technique also allocates decision levels densely in the small-magnitude region, which corresponds to a high occurrence probability, but in a different way. We conclude that nonuniform quantization achieves minimum mean square quantization error by distributing decision levels according to the statistics of the input random variable.

These two types of nonuniform quantizers are both time-invariant. That is, they are not designed for nonstationary input signals. Moreover, even for a stationary input signal, if its *pdf* deviates from that with which the optimum quantizer is designed, then what is called *mismatch* will take place and the performance of the quantizer will deteriorate. There are two main types of mismatch. One is called variance mismatch. That is, the *pdf* of input signal is matched, while the variance is mismatched. Another type is *pdf* mismatch. Noted that these two kinds of mismatch also occur in optimum uniform quantization, since there the optimization is also achieved based on the input statistics assumption. For a detailed analysis of the effects of the two types of mismatch on quantization, readers are referred to (Jayant and Noll, 1984).

Adaptive quantization attempts to make the quantizer design adapt to the varying input statistics in order to achieve better performance. It is a means to combat the mismatch problem discussed above. By statistics, we mean the statistical mean, variance (or the dynamic range), and type of input *pdf*. When the mean of the input changes, differential coding (discussed in the next chapter) is a suitable method to handle the variation. For other types of cases, adaptive quantization is found to be effective. The price paid for adaptive quantization is processing delays and an extra storage requirement as seen below.

There are two different types of adaptive quantization: forward adaptation and backward adaptation. Before we discuss these, however, let us describe an alternative way to define quantization (Jayant and Noll, 1984). Look at Figure 2.14. Quantization can be viewed as a two-stage

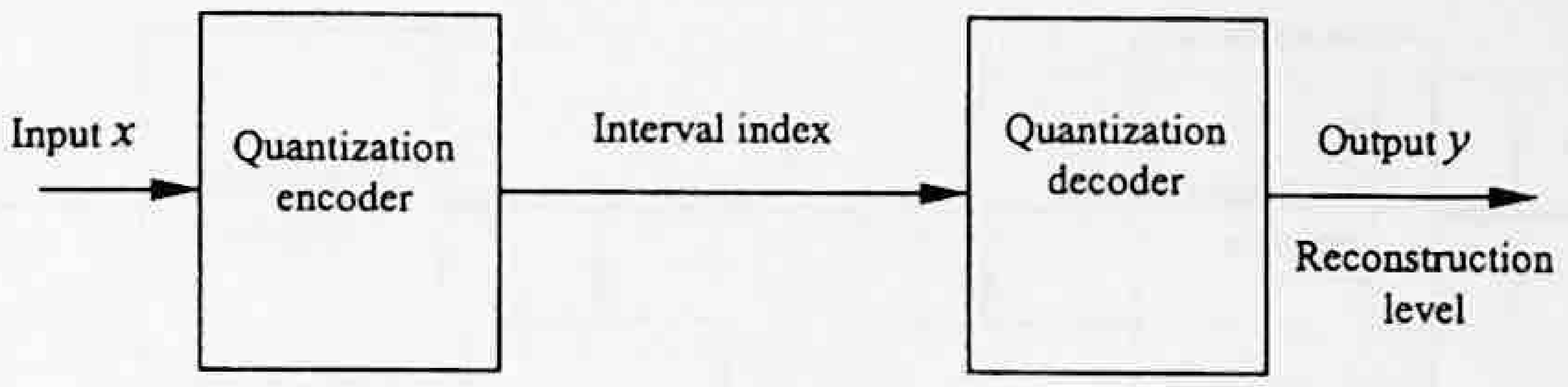


FIGURE 2.14 A two-stage model of quantization.

process. The first stage is the quantization encoder and the second stage is the quantization decoder. In the encoder, the input to quantization is converted to the index of an interval into which the input x falls. This index is mapped to (the codeword that represents) the reconstruction level corresponding to the interval in the decoder. Roughly speaking, this definition considers a quantizer as a communication system in which the quantization encoder is on the transmitter side while the quantization decoder is on the receiver side. In this sense, this definition is broader than that for quantization defined in Figure 2.3(a).

2.4.1 Forward Adaptive Quantization

A block diagram of forward adaptive quantization is shown in Figure 2.15. There, the input to the quantizer, x, is first split into blocks, each with a certain length. Blocks are stored in a buffer one at a time. A statistical analysis is then carried out with respect to the block in the buffer. Based on the analysis, the quantization encoder is set up, and the input data within the block are assigned indexes of respective intervals. In addition to these indexes, the encoder setting parameters, derived from the statistical analysis, are sent to the quantization decoder as *side* information. The term *side* comes from the fact that the amount of bits used for coding the setting parameter is usually a small fraction of the total amount of bits used.

Selection of the block size is a critical issue. If the size is small, the adaptation to the local statistics will be effective, but the side information needs to be sent frequently. That is, more bits are used for sending the side information. If the size is large, the bits used for side information decrease. On the other hand, the adaptation becomes less sensitive to changing statistics, and both processing delays and storage required increase. In practice, a proper compromise between the quantity of side information and the effectiveness of adaptation produces a good selection of the block size.

Examples of using the forward approach to adapt quantization to a changing input variance (to combat variance mismatch) can be found in (Jayant and Noll, 1984; Sayood, 1996).

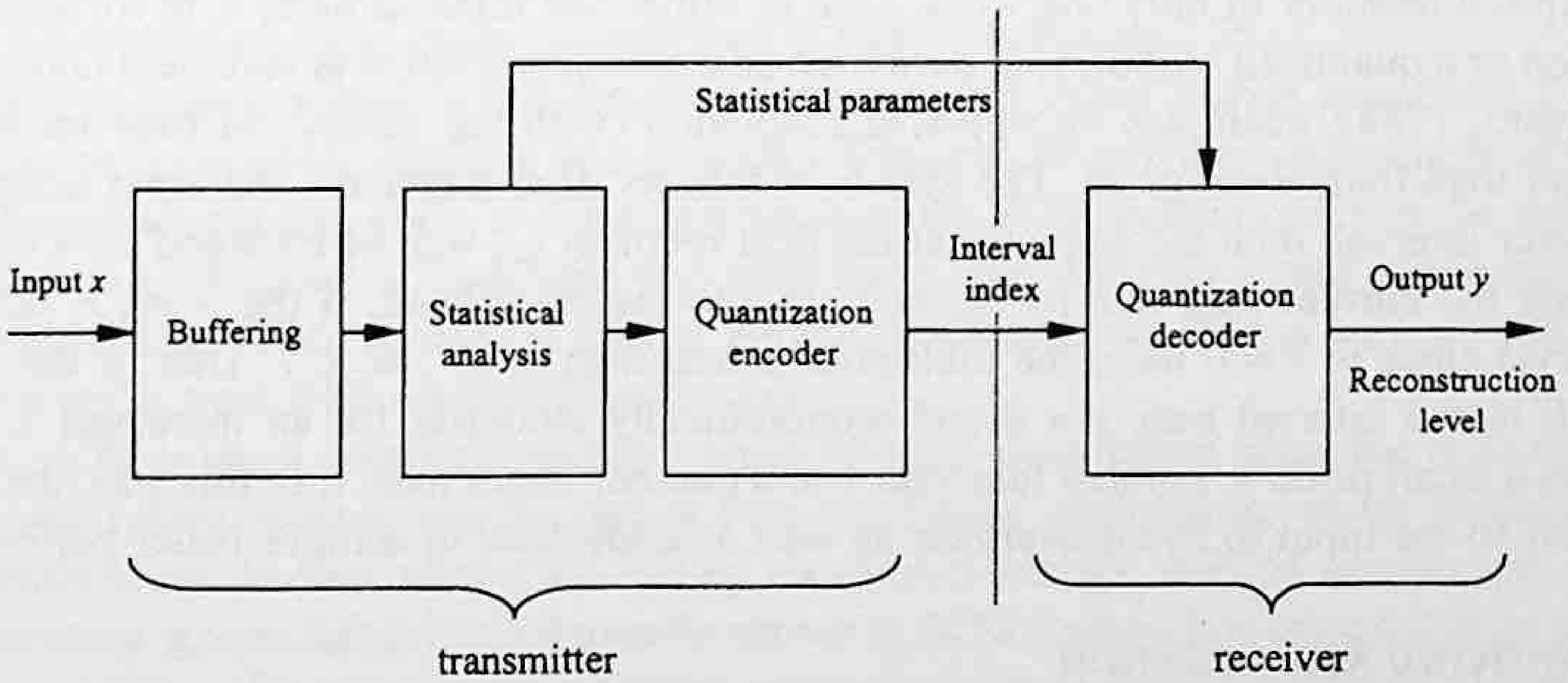


FIGURE 2.15 Forward adaptive quantization.

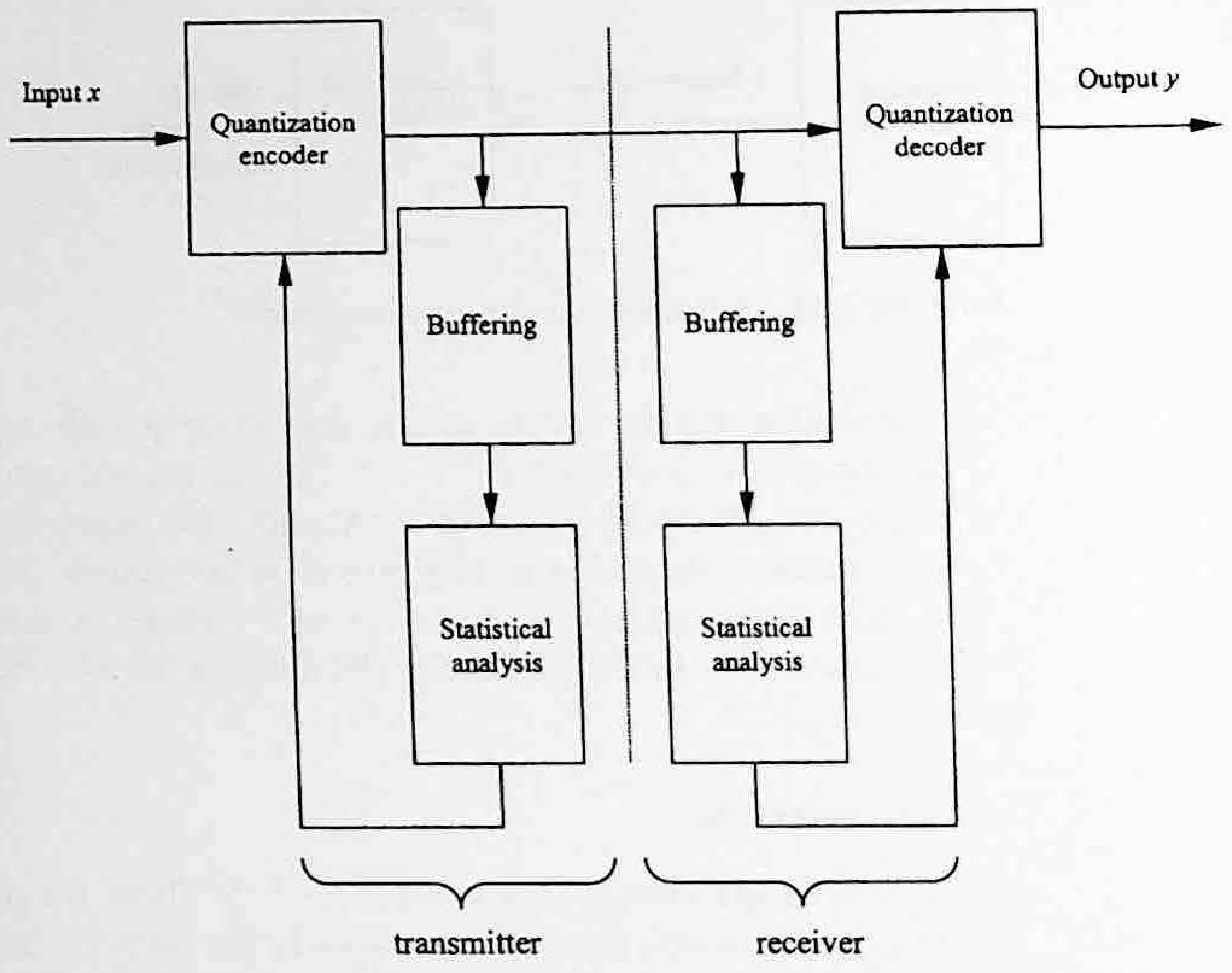


FIGURE 2.16 Backward adaptive quantization.

2.4.2 Backward Adaptive Quantization

Figure 2.16 shows a block diagram of backward adaptive quantization. A close look at the block diagram reveals that in both the quantization encoder and decoder the buffering and the statistical analysis are carried out with respect to the output of the quantization encoder. In this way, there is no need to send side information. The sensitivity of adaptation to the changing statistics will be degraded, however, since instead of the original input, only the output of the quantization encoder is used in the statistical analysis. That is, the quantization noise is involved in the statistical analysis.

2.4.3 Adaptive Quantization with a One-Word Memory

Intuitively, it is expected that observance of a sufficiently large number of input or output (quantized) data is necessary in order to track the changing statistics and then adapt the quantizer setting in adaptive quantization. Through an analysis, Jayant showed that effective adaptations can be realized with an explicit memory of only one word. That is, either one input sample, x, in forward adaptive quantization or a quantized output, y, in backward adaptive quantization is sufficient (Jayant, 1973).

In (Jayant, 1984), examples on step-size adaptation (with the number of total reconstruction levels larger than four) were given. The idea is as follows. If at moment t_i the input sample x_i falls into the outer interval, then the step size at the next moment t_{i+1} will be enlarged by a factor of m_i (multiplying the current step size by m_i, $m_i > 1$). On the other hand, if the input x_i falls into an inner interval close to $x = 0$ then, the multiplier is less than 1, i.e., $m_i < 1$. That is, the multiplier m_i is small in the interval near $x = 0$ and monotonically increases for an increased x. Its range varies from a small positive number less than 1 to a number larger than 1. In this way, the quantizer adapts itself to the input to avoid *overload* as well as *underload* to achieve better performance.

2.4.4 Switched Quantization

This is another adaptive quantization scheme. A block diagram is shown in Figure 2.17. It consists of a bank of L quantizers. Each quantizer in the bank is fixed, but collectively they form a bank

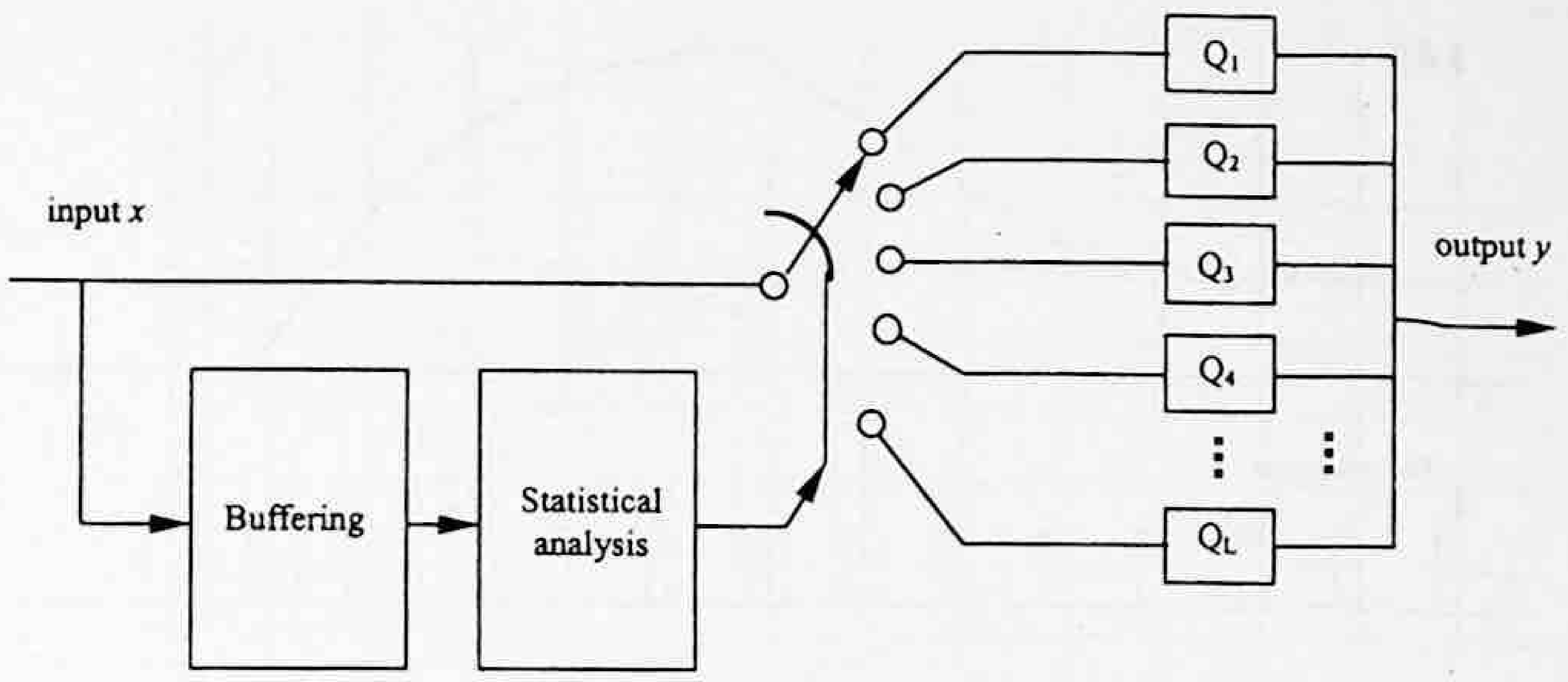


FIGURE 2.17 Switched quantization.

of quantizers with a variety of input-output characteristics. Based on a statistical analysis of recent input or output samples, a switch connects the current input to one of the quantizers in the bank such that the best possible performance may be achieved. It is reported that in both video and speech applications, this scheme has shown improved performance even when the number of quantizers in the bank, L, is two (Jayant and Noll, 1984). Interestingly, it is noted that as $L \rightarrow \infty$, the switched quantization converges to the adaptive quantizer discussed above.

2.5 PCM

Pulse code modulation (PCM) is closely related to quantization, the focus of this chapter. Furthermore, as pointed out in (Jayant, 1984), PCM is the earliest, best established, and most frequently applied coding system despite the fact that it is the most bit-consuming digitizing system (since it encodes each pixel independently) as well as being a very demanding system in terms of the bit error rate on the digital channel. Therefore, we discuss the PCM technique in this section.

PCM is now the most important form of pulse modulation. The other forms of pulse modulation are pulse amplitude modulation (PAM), pulse width modulation (PWM), and pulse position modulation (PPM), which are covered in most communications texts. Briefly speaking, pulse modulation links an analog signal to a pulse train in the following way. The analog signal is first sampled (a discretization in the time domain). The sampled values are used to modulate a pulse train. If the modulation is carried out through the amplitude of the pulse train, it is called PAM. If the modified parameter of the pulse train is the pulse width, we then have PWM. If the pulse width and magnitude are constant — only the position of pulses is modulated by the sample values — we then encounter PPM. An illustration of these pulse modulations is shown in Figure 2.18.

In PCM, an analog signal is first sampled. The sampled value is then quantized. Finally the quantized value is encoded, resulting in a bit steam. Figure 2.19 provides an example of PCM. We see that through a sampling and a uniform quantization the PCM system converts the input analog signal, which is continuous in both time and magnitude, into a digital signal (discretized in both time and magnitude) in the form of a natural binary code sequence. In this way, an analog signal modulates a pulse train with a natural binary code.

By far, PCM is more popular than other types of pulse modulation since the *code* modulation is much more robust against various noises than amplitude modulation, width modulation, and position modulation. In fact, almost all coding techniques include a PCM component. In digital image processing, given digital images usually appear in PCM format. It is known that an acceptable PCM representation of a monochrome picture requires six to eight bits per pixel (Huang, 1975). It is used so commonly in practice that its performance normally serves as a standard against which other coding techniques are compared.

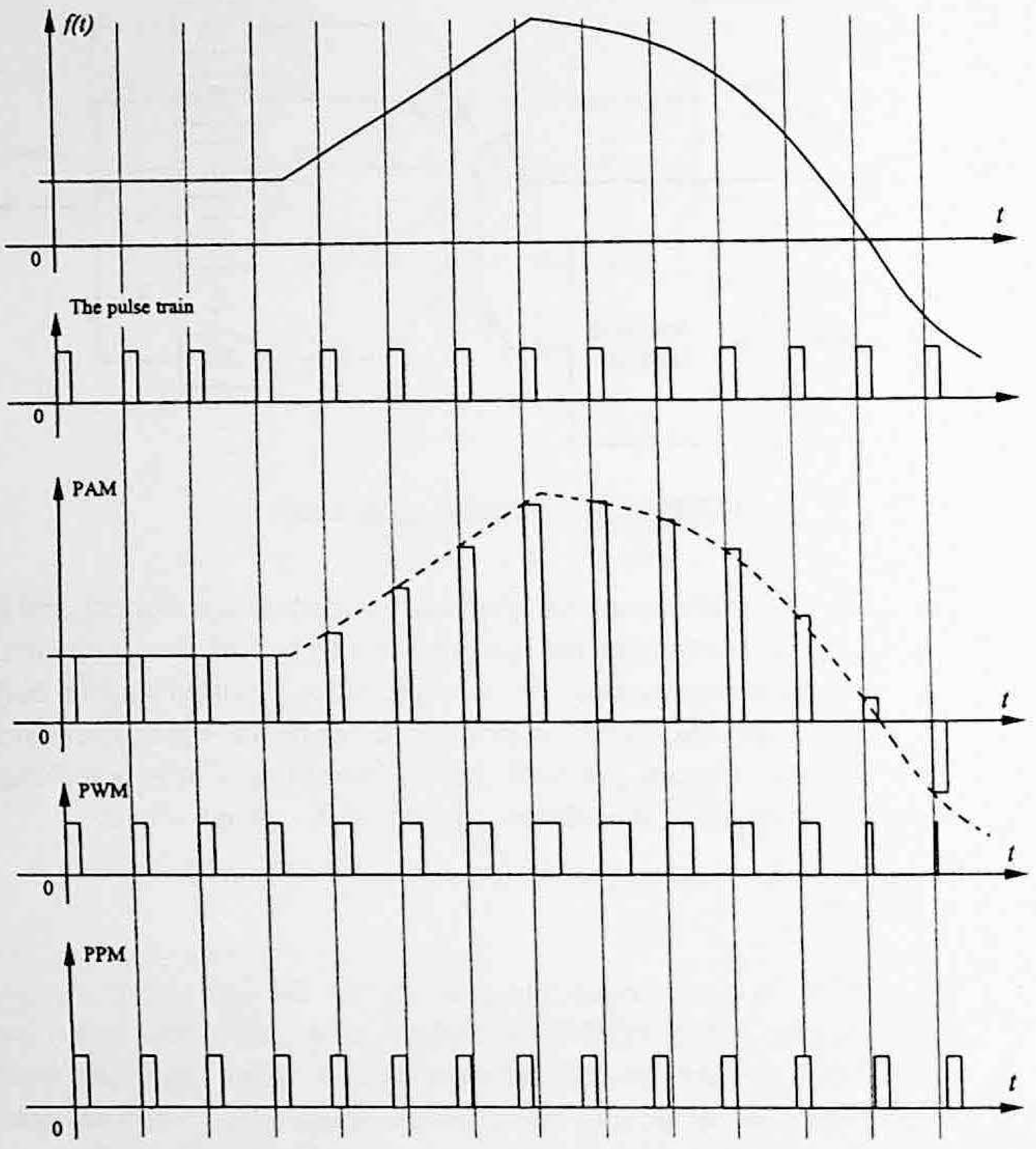


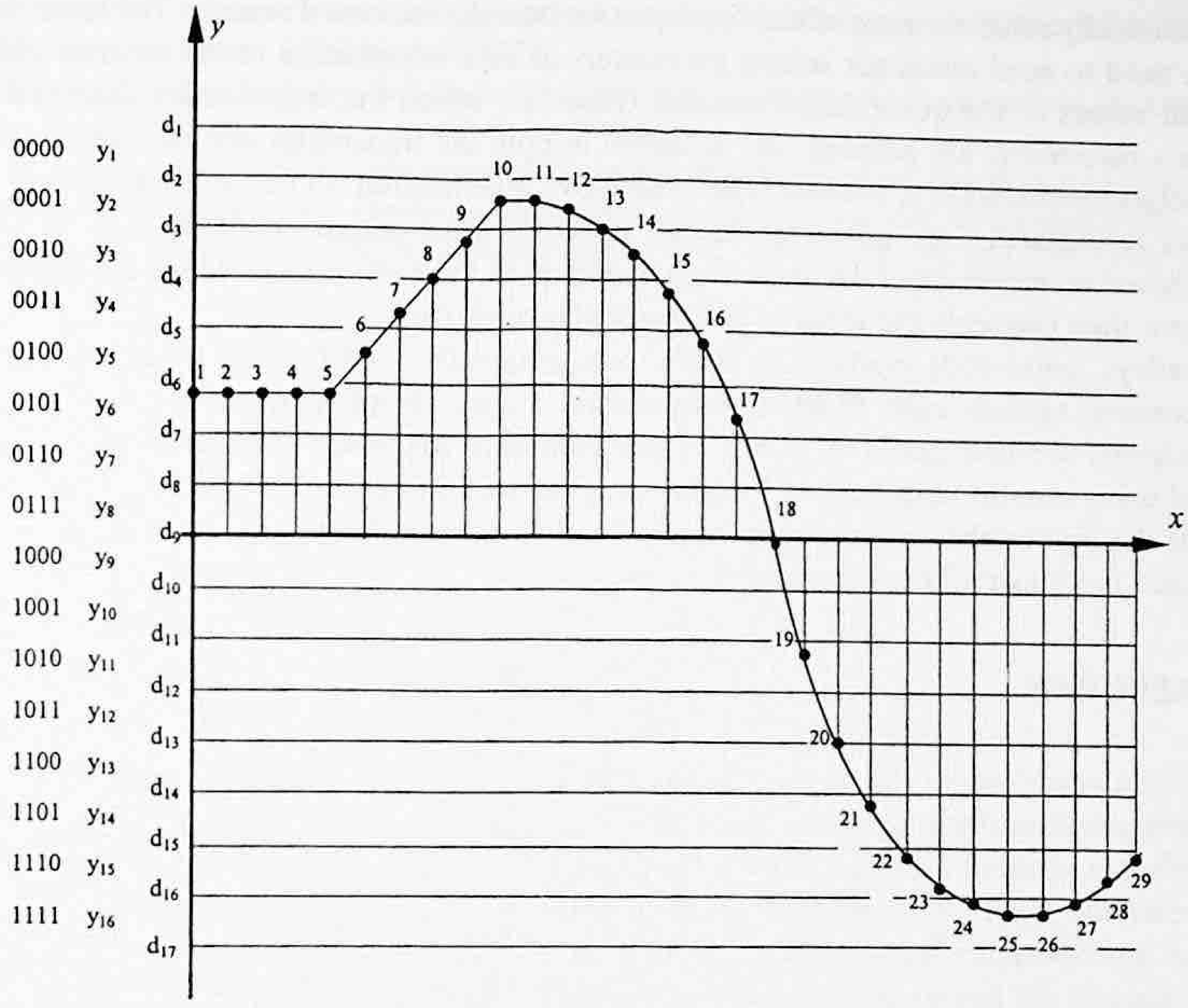
FIGURE 2.18 Pulse modulation.

Recall the false contouring phenomenon discussed in Chapter 1, when we discussed texture masking. It states that our eyes are more sensitive to relatively uniform regions in an image plane. If the number of reconstruction levels is not large enough (coarse quantization), then some unnatural contours will appear. When frequency masking was discussed, it was noted that by adding some high-frequency signal before quantization, the false contouring can be eliminated to a great extent. This technique is called dithering. The high-frequency signal used is referred to as a dither signal. Both false contouring and dithering were first reported in (Goodall, 1951).

2.6 SUMMARY

Quantization is a process in which a quantity having possibly an infinite number of different values is converted to another quantity having only finite many values. It is an important element in source encoding that has significant impact on both bit rate and distortion of reconstructed images and video in visual communication systems. Depending on whether the quantity is a scalar or a vector, quantization is called either scalar quantization or vector quantization. In this chapter we considered only scalar quantization.

Uniform quantization is the simplest and yet the most important case. In uniform quantization, except for outer intervals, both decision levels and reconstruction levels are uniformly spaced. Moreover, a reconstruction level is the arithmetic average of the two corresponding decision levels. In uniform quantization design, the step size is usually the only parameter that needs to be specified.



Output code (from left to right, from top to bottom):

0101	0101	0101	0101	0101	0100	0011	0011	0010	0001	0001	0001	0010	0010	0011
0100	0101	1000	1010	1100	1101	1110	1110	1111	1111	1111	1111	1110	1110	

FIGURE 2.19 Pulse code modulation (PCM).

Optimum quantization implies minimization of the mean square quantization error. When the input has a uniform distribution, uniform quantization is optimum. For the sake of simplicity, a uniform optimum quantizer is sometimes desired even when the input does not obey uniform distribution. The design under these circumstances involves an iterative procedure. The design problem in cases where the input has Gaussian, Lapacian, or Gamma distribution was solved and the parameters are available.

When the constraint of uniform quantization is removed, the conditions for optimum quantization are derived. The resultant optimum quantizer is normally nonuniform. An iterative procedure to solve the design is established and the optimum design parameters for Gaussian, Laplacian, and Gamma distribution are tabulated.

The companding technique is an alternative way to implement nonuniform quantization. Both nonuniform quantization and companding are time-invariant and hence not suitable for nonstationary input. Adaptive quantization deals with nonstationary input and combats the mismatch that occurs in optimum quantization design.

In adaptive quantization, buffering is necessary to store some recent input or sampled output data. A statistical analysis is carried out with respect to the stored recent data. Based on the analysis, the quantizer's parameters are adapted to changing input statistics to achieve better quantization performance. There are two types of adaptive quantization: forward and backward adaptive quantization. With the forward type, the statistical analysis is derived from the original input data, while with the backward type, quantization noise is involved in the analysis. Therefore, the forward technique usually achieves more effective adaptation than the backward manner. The latter, however, does not need to send quantizer setting parameters as side information to the receiver side, since the output values of the quantization encoder (based on which the statistics are analyzed and the quantizer's parameters are adapted) are available in both the transmitter and receiver sides.

Switched quantization is another type of adaptive quantization. In this scheme, a bank of fixed quantizers is utilized, each quantizer having different input-output characteristics. A statistical analysis based on recent input decides which quantizer in the bank is suitable for the present input. The system then connects the input to this particular quantizer.

Nowadays, pulse code modulation is the most frequently used form of pulse modulation due to its robustness against noise. PCM consists of three stages: sampling, quantization, and encoding. Analog signals are first sampled with a proper sampling frequency. The sampled data are then quantized using a uniform quantizer. Finally, the quantized values are encoded with natural binary code. It is the best established and most applied coding system. Despite its bit-consuming feature, it is utilized in almost all coding systems.

2.7 EXERCISES

2-1. Using your own words, define quantization and uniform quantization. What are the two features of uniform quantization?

2-2. What is optimum quantization? Why is uniform quantization sometimes desired, even when the input has a *pdf* different from uniform? How was this problem solved? Draw an input-output characteristic of an optimum uniform quantizer with an input obeying Gaussian *pdf* having zero mean, unit variance, and the number of reconstruction levels, N, equal to 8.

2-3. What are the conditions of optimum nonuniform quantization? From Table 2.2, what observations can you make?

2-4. Define variance mismatch and *pdf* mismatch. Discuss how you can resolve the mismatch problem.

2-5. What is the difference between forward and backward adaptive quantization? Comment on the merits and drawbacks for each.

2-6. What are PAM, PWM, PPM, and PCM? Why is PCM the most popular type of pulse modulation?

REFERENCES

Fleischer, P. E. Sufficient conditions for achieving minimum distortion in quantizer, *IEEE Int. Convention Rec.*, 12, 104-111, 1964.

Gersho, A, Quantization, *IEEE Commun. Mag.*, pp. 6-29, September, 1977.

Gonzalez, R. C. and R. E. Woods, *Digital Image Processing*, Addison-Wesley, Reading, MA, 1992.

Goodall, W. M. Television by pulse code modulation, *Bell Syst. Tech. J.*, 33-49, January 1951.

Huang, T. S. PCM picture transmission, *IEEE Spectrum*, 2, 57-63, 1965.

Jayant, N. S. Adaptive delta modulation with one-bit memory, *Bell Syst. Tech. J.*, 49, 321-342, 1970.

Jayant, N. S. Adaptive quantization with one word memory, *Bell Syst. Tech. J.*, 52, 1119-1144, 1973.

Jayant, N. S. and P. Noll, *Digital Coding of Waveforms*, Prentice-Hall, Englewood Cliffs, NJ, 1984.

Li, W. and Y.-Q. Zhang, Vector-based signal processing and qunatization for image and video compression, *Proc. IEEE*, 83(2), 317-335, 1995.

Lloyd, S. P. Least squares quantization in PCM, Inst. Mathematical Statistics Meet., Atlantic City, NJ, September 1957; *IEEE Trans. Inf. Theory*, pp. 129-136, March 1982.

Max, J. Quantizing for minimum distortion, *IRE Trans. Inf. Theory*, it-6, 7-12, 1960.

Musmann, H. G. Predictive Image Coding, in *Image Transmission Techniques*, W. K. Pratt (Ed.), Academic Press, New York, 1979.

Paez, M. D. and T. H. Glisson, Minimum mean squared error qunatization in speech PCM and DPCM Systems, *IEEE Trans. Commun.*, 225-230, April 1972.

Panter, P. F. and W. Dite, Quantization distortion in pulse count modulation with nonuniform spacing of levels, *Proc. IRE*, 39, 44-48, 1951.

Sayood, K. *Introduction to Data Compression*, Morgan Kaufmann Publishers, San Francisco, CA, 1996.

Sklar, B. *Digital Communications: Fundamentals and Applications*, Prentice-Hall, Englewood Cliffs, NJ, 1988.

Smith, B. Instantaneous companding of quantized signals, *Bell Syst. Tech. J.*, 36, 653-709, 1957.

3 Differential Coding

Instead of encoding a signal directly, the *differential coding* technique codes the difference between the signal itself and its prediction. Therefore it is also known as *predictive coding*. By utilizing spatial and/or temporal interpixel correlation, differential coding is an efficient and yet computationally simple coding technique. In this chapter, we first describe the differential technique in general. Two components of differential coding, prediction and quantization, are discussed. There is an emphasis on (optimum) prediction, since quantization was discussed in Chapter 2. When the difference signal (also known as prediction error) is quantized, the differential coding is called differential pulse code modulation (DPCM). Some issues in DPCM are discussed, after which delta modulation (DM) as a special case of DPCM is covered. The idea of differential coding involving image sequences is briefly discussed in this chapter. More detailed coverage is presented in Sections III and IV, starting from Chapter 10. If quantization is not included, the differential coding is referred to as information-preserving differential coding. This is discussed at the end of the chapter.

3.1 INTRODUCTION TO DPCM

As depicted in Figure 2.3, a source encoder consists of the following three components: transformation, quantization, and codeword assignment. The transformation converts input into a format for quantization followed by codeword assignment. In other words, the component of transformation decides which format of input is to be encoded. As mentioned in the previous chapter, input itself is not necessarily the most suitable format for encoding.

Consider the case of monochrome image encoding. The input is usually a 2-D array of gray level values of an image obtained via PCM coding. The concept of spatial redundancy, discussed in Section 1.2.1.1, tells us that neighboring pixels of an image are usually highly correlated. Therefore, it is more efficient to encode the gray difference between two neighboring pixels instead of encoding the gray level values of each pixel. At the receiver, the decoded difference is added back to reconstruct the gray level value of the pixel. Since neighboring pixels are highly correlated, their gray level values bear a great similarity. Hence, we expect that the variance of the difference signal will be smaller than that of the original signal. Assume uniform quantization and natural binary coding for the sake of simplicity. Then we see that for the same bit rate (bits per sample) the quantization error will be smaller, i.e., a higher quality of reconstructed signal can be achieved. Or, for the same quality of reconstructed signal, we need a lower bit rate.

3.1.1 SIMPLE PIXEL-TO-PIXEL DPCM

Denote the gray level values of pixels along a row of an image as z_i, $i = 1,\cdots,M$, where M is the total number of pixels within the row. Using the immediately preceding pixel's gray level value, z_{i-1}, as a prediction of that of the present pixel, $\hat{z}_i$, i.e.,

$$\hat{z}_i = z_{i-1} \tag{3.1}$$

we then have the difference signal

$$d_i = z_i - \hat{z}_i = z_i - z_{i-1} \tag{3.2}$$

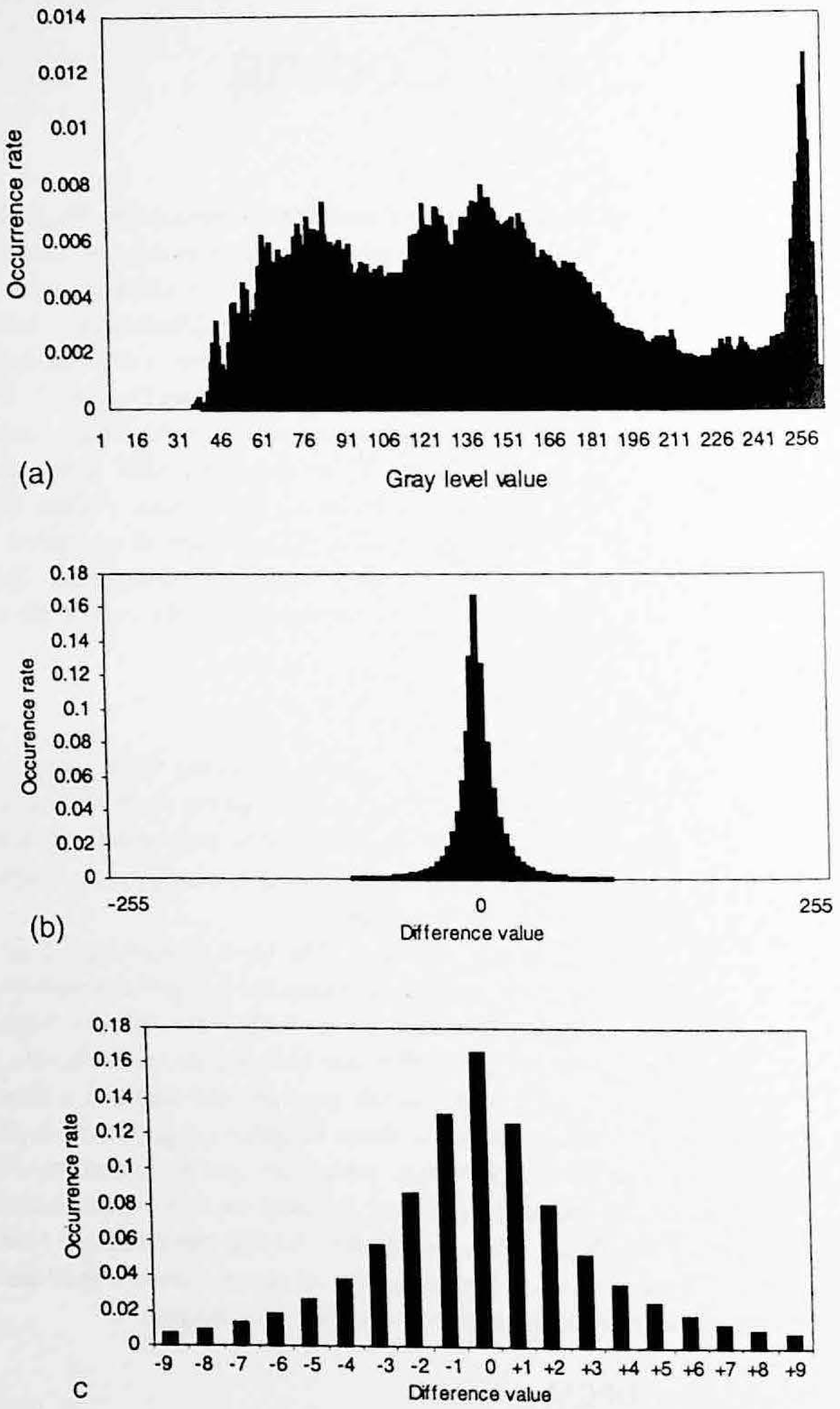


FIGURE 3.1 (a) Histogram of the original "boy and girl" image. (b) Histogram of the difference image obtained by using horizontal pixel-to-pixel differencing. (c) A close-up of the central portion of the histogram of the difference image.

Assume a bit rate of eight bits per sample in the quantization. We can see that although the dynamic range of the difference signal is theoretically doubled, from 256 to 512, the variance of the difference signal is actually much smaller. This can be confirmed from the histograms of the "boy and girl" image (refer to Figure 1.1) and its difference image obtained by horizontal pixel-to-pixel differencing, shown in Figure 3.1(a) and (b), respectively. Figure 3.1(b) and its close-up (c) indicate that by a rate of 42.44% the difference values fall into the range of −1, 0, and +1. In other words, the histogram of the difference signal is much more narrowly concentrated than that of the original signal.

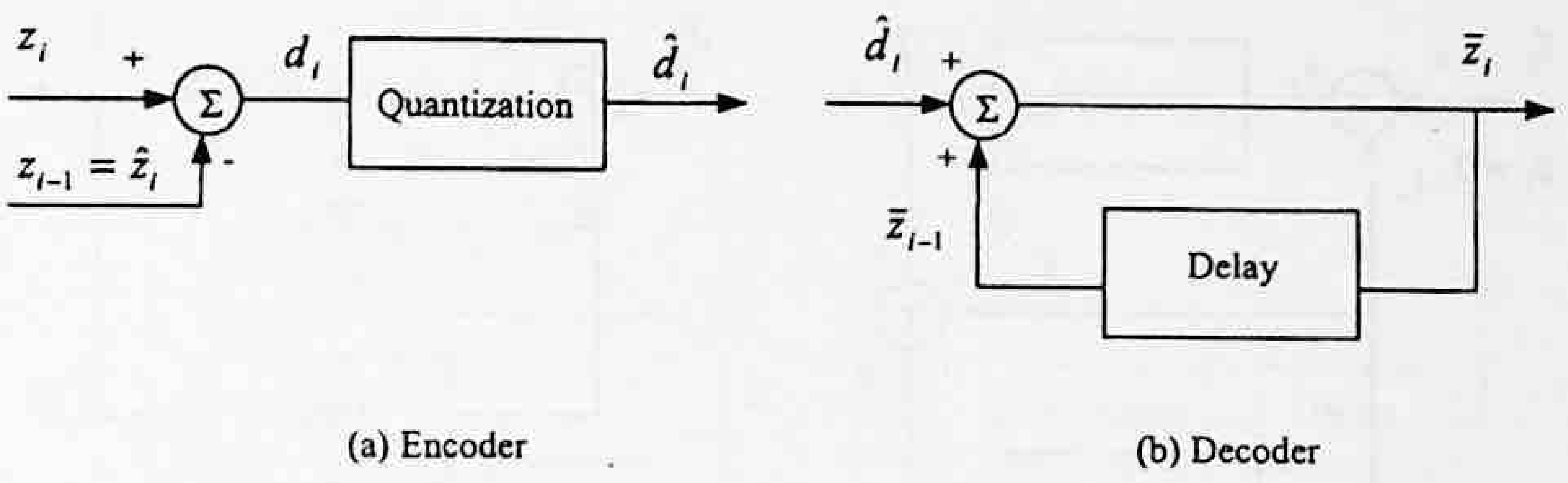


FIGURE 3.2 Block diagram of a pixel-to-pixel differential coding system.

A block diagram of the scheme described above is shown in Figure 3.2. There z_i denotes the sequence of pixels along a row, d_i is the corresponding difference signal, and $\hat{d}_i$ is the quantized version of the difference, i.e.,

$$\hat{d}_i = Q(d_i) = d_i + e_q \tag{3.3}$$

where e_q represents the quantization error. In the decoder, $\bar{z}_i$ represents the reconstructed pixel gray value, and we have

$$\bar{z}_i = \bar{z}_{i-1} + \hat{d}_i \tag{3.4}$$

This simple scheme, however, suffers from an accumulated quantization error. We can see this clearly from the following derivation (Sayood, 1996), where we assume the initial value z_0 is available for both the encoder and the decoder.

$$\begin{aligned} \text{as} \quad i = 1, \quad d_1 &= z_1 - z_0 \\ \hat{d}_1 &= d_1 + e_{q,1} \\ \bar{z}_1 = z_0 + \hat{d}_1 &= z_0 + d_1 + e_{q,1} = z_1 + e_{q,1} \end{aligned} \tag{3.5}$$

Similarly, we can have

$$as \quad i = 2, \quad \bar{z}_2 = z_2 + e_{q,1} + e_{q,2} \tag{3.6}$$

and, in general,

$$\bar{z}_i = z_i + \sum_{j=1}^{i} e_{q,j} \tag{3.7}$$

This problem can be remedied by the following scheme, shown in Figure 3.3. Now we see that in both the encoder and the decoder, the reconstructed signal is generated in the same way, i.e.,

$$\bar{z}_i = \bar{z}_{i-1} + \hat{d}_i \tag{3.8}$$

and in the encoder the difference signal changes to

$$d_i = z_i - \bar{z}_{i-1} \tag{3.9}$$

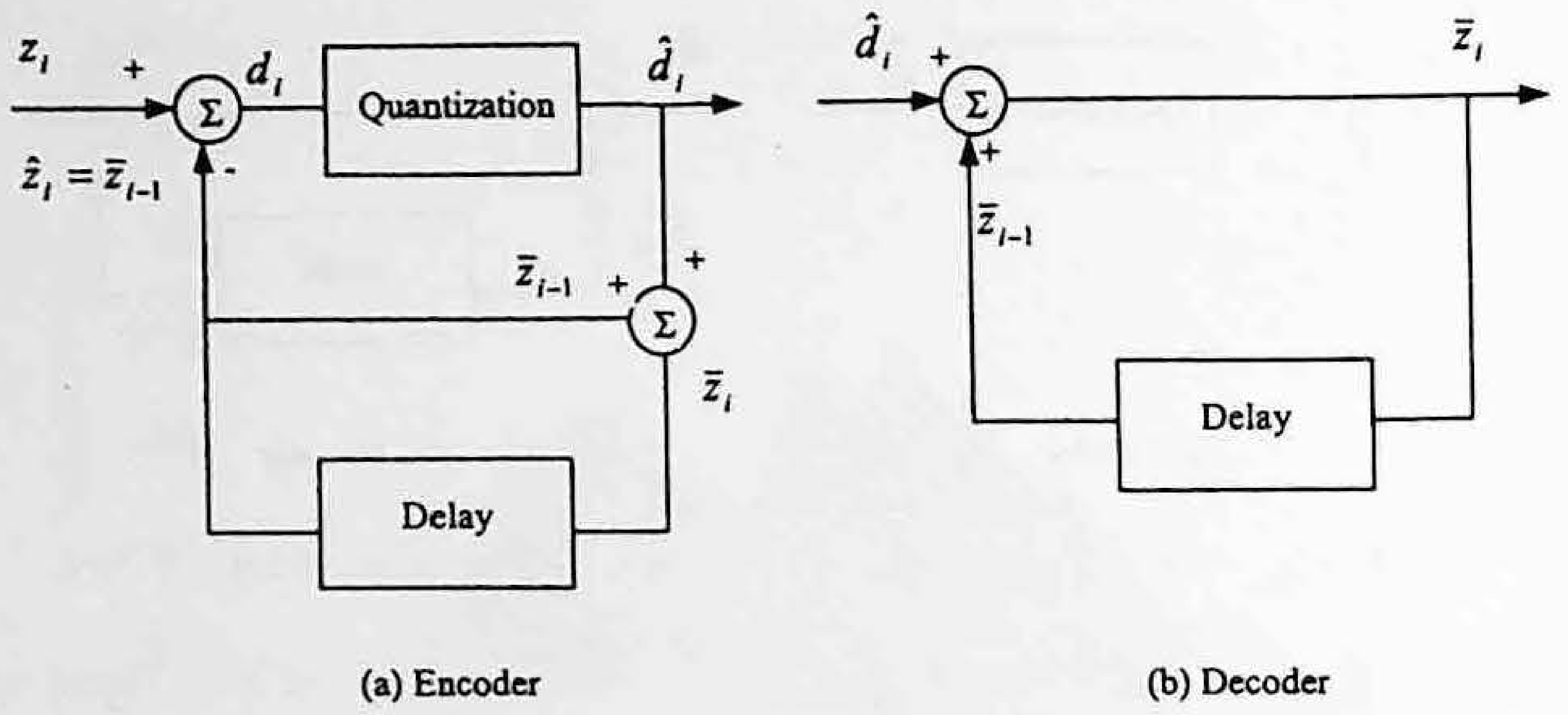


FIGURE 3.3 Block diagram of a practical pixel-to-pixel differential coding system.

Thus, the previously reconstructed $\bar{z}_{i-1}$ is used as the predictor, $\hat{z}_i$, i.e.,

$$\hat{z}_i = \bar{z}_{i-1}. \tag{3.10}$$

In this way, we have

$$\begin{aligned} as \quad i = 1, \quad d_1 &= z_1 - z_0 \\ \hat{d}_1 &= d_1 + e_{q,1} \\ \bar{z}_1 = z_0 + \hat{d}_1 &= z_0 + d_1 + e_{q,1} = z_1 + e_{q,1} \end{aligned} \tag{3.11}$$

Similarly, we have

$$\begin{aligned} as \quad i = 2, \quad d_2 &= z_2 - \bar{z}_1 \\ \hat{d}_2 &= d_2 + e_{q,2} \\ \bar{z}_2 = \bar{z}_1 + \hat{d}_2 &= z_2 + e_{q,2} \end{aligned} \tag{3.12}$$

In general,

$$\bar{z}_i = z_i + e_{q,i} \tag{3.13}$$

Thus, we see that the problem of the quantization error accumulation has been resolved by having both the encoder and the decoder work in the same fashion, as indicated in Figure 3.3, or in Equations 3.3, 3.9, and 3.10.

3.1.2 General DPCM Systems

In the above discussion, we can view the reconstructed neighboring pixel's gray value as a prediction of that of the pixel being coded. Now, we generalize this simple pixel-to-pixel DPCM. In a general DPCM system, a pixel's gray level value is first predicted from the preceding reconstructed pixels' gray level values. The difference between the pixel's gray level value and the predicted value is then quantized. Finally, the quantized difference is encoded and transmitted to the receiver. A block

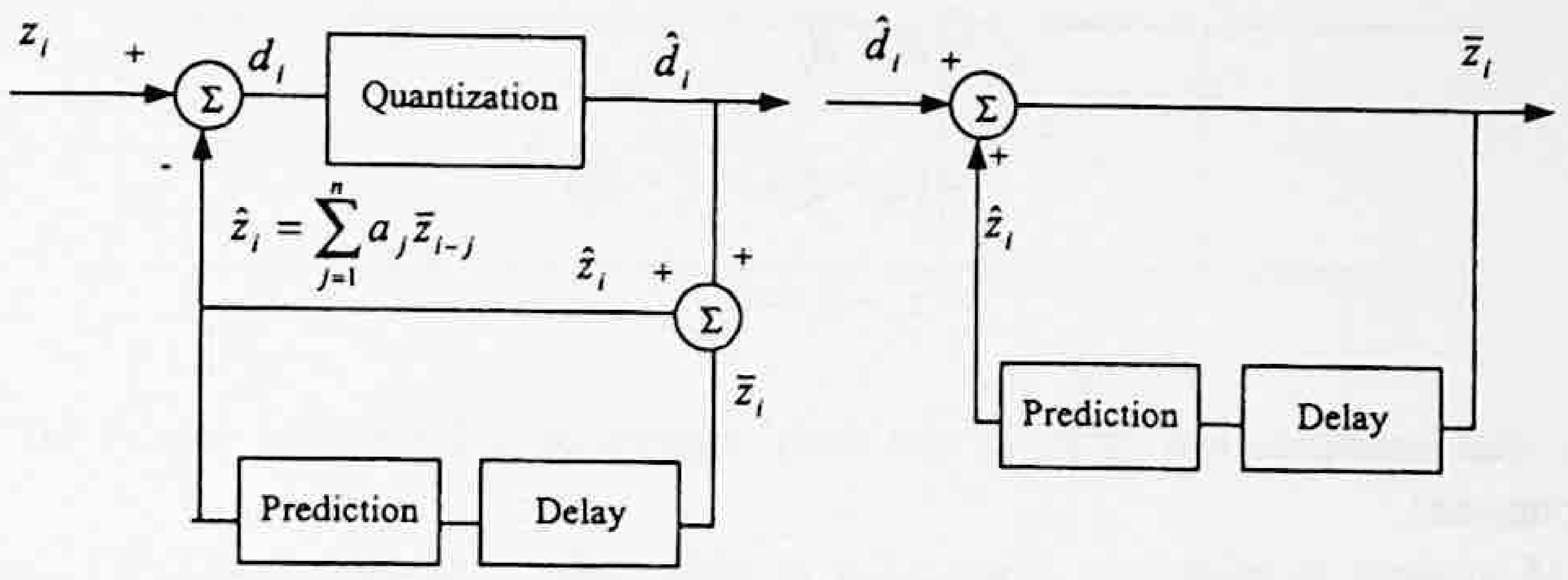


FIGURE 3.4 Block diagram of a general DPCM system.

diagram of this general differential coding scheme is shown in Figure 3.4, where the codeword assignment in the encoder and its counterpart in decoder are not included.

It is noted that, instead of using the previously reconstructed sample, $\bar{z}_{i-1}$, as a predictor, we now have the predicted version of z_i, $\hat{z}_i$, as a function of the n previously reconstructed samples, $\bar{z}_{i-1}, \bar{z}_{i-2}, \cdots, \bar{z}_{i-n}$. That is,

$$\hat{z}_i = f\left(\bar{z}_{i-1}, \bar{z}_{i-2}, \cdots, \bar{z}_{i-n}\right) \tag{3.14}$$

Linear prediction, i.e., that the function f in Equation 3.14 is linear, is of particular interest and is widely used in differential coding. In linear prediction, we have

$$\hat{z}_i = \sum_{j=1}^{n} a_j \bar{z}_{i-j} \tag{3.15}$$

where a_j are real parameters. Hence, we see that the simple pixel-to-pixel differential coding is a special case of general differential coding with linear prediction, i.e., $n = 1$ and $a_1 = 1$.

In Figure 3.4, d_i is the difference signal and is equal to the difference between the original signal, z_i, and the prediction $\hat{z}_i$. That is,

$$d_i = z_i - \hat{z}_i \tag{3.16}$$

The quantized version of d_i is denoted by $\hat{d}_i$. The reconstructed version of z_i is represented by $\bar{z}_i$, and

$$\bar{z}_i = \hat{z}_i + \hat{d}_i \tag{3.17}$$

Note that this is true for both the encoder and the decoder. Recall that the accumulation of the quantization error can be remedied by using this method.

The difference between the original input and the predicted input is called prediction error, which is denoted by e_p. That is,

$$e_p = z_i - \hat{z}_i \tag{3.18}$$

where the e_p is understood as the prediction error associated with the index i. Quantization error, e_q, is equal to the reconstruction error or coding error, e_r, defined as the difference between the original signal, z_i, and the reconstructed signal, $\bar{z}_i$, when the transmission is error free:

$$\begin{aligned} e_q &= d_i - \hat{d}_i \\ &= (z_i - \hat{z}_i) - (\bar{z}_i - \hat{z}_i) \\ &= z_i - \bar{z}_i = e_r \end{aligned} \tag{3.19}$$

This indicates that quantization error is the only source of information loss with an error-free transmission channel.

The DPCM system depicted in Figure 3.4 is also called closed-loop DPCM with feedback around the quantizer (Jayant, 1984). This term reflects the feature in DPCM structure.

Before we leave this section, let us take a look at the history of the development of differential image coding. According to an excellent early article on differential image coding (Musmann, 1979), the first theoretical and experimental approaches to image coding involving linear prediction began in 1952 at the Bell Telephone Laboratories (Oliver, 1952; Kretzmer, 1952; Harrison, 1952). The concepts of DPCM and DM were also developed in 1952 (Cutler, 1952; Dejager, 1952). Predictive coding capable of preserving information for a PCM signal was established at the Massachusetts Institute of Technology (Elias, 1955).

The differential coding technique has played an important role in image and video coding. In the international coding standard for still images, JPEG (covered in Chapter 7), we can see that differential coding is used in the lossless mode and in the DCT-based mode for coding DC coefficients. Motion-compensated (MC) coding has been a major development in video coding since the 1980s and has been adopted by all the international video coding standards such as H.261 and H.263 (covered in Chapter 19), MPEG 1 and MPEG 2 (covered in Chapter 16). MC coding is essentially a predictive coding technique applied to video sequences involving displacement motion vectors.

3.2 OPTIMUM LINEAR PREDICTION

Figure 3.4 demonstrates that a differential coding system consists of two major components: prediction and quantization. Quantization was discussed in the previous chapter. Hence, in this chapter we emphasize prediction. Below, we formulate an optimum linear prediction problem and then present a theoretical solution to the problem.

3.2.1 FORMULATION

Optimum linear prediction can be formulated as follows. Consider a discrete-time random process z. At a typical moment i, it is a random variable z_i. We have n previous observations $\bar{z}_{i-1}, \bar{z}_{i-2}, \cdots, \bar{z}_{i-n}$ available and would like to form a prediction of z_i, denoted by $\hat{z}_i$. The output of the predictor, $\hat{z}_i$, is a linear function of the n previous observations. That is,

$$\hat{z}_i = \sum_{j=1}^{n} a_j \bar{z}_{i-j} \tag{3.20}$$

with a_j, $j = 1,2,\cdots,n$ being a set of real coefficients. An illustration of a linear predictor is shown in Figure 3.5. As defined above, the prediction error, e_p, is

$$e_p = z_i - \hat{z}_i \tag{3.21}$$

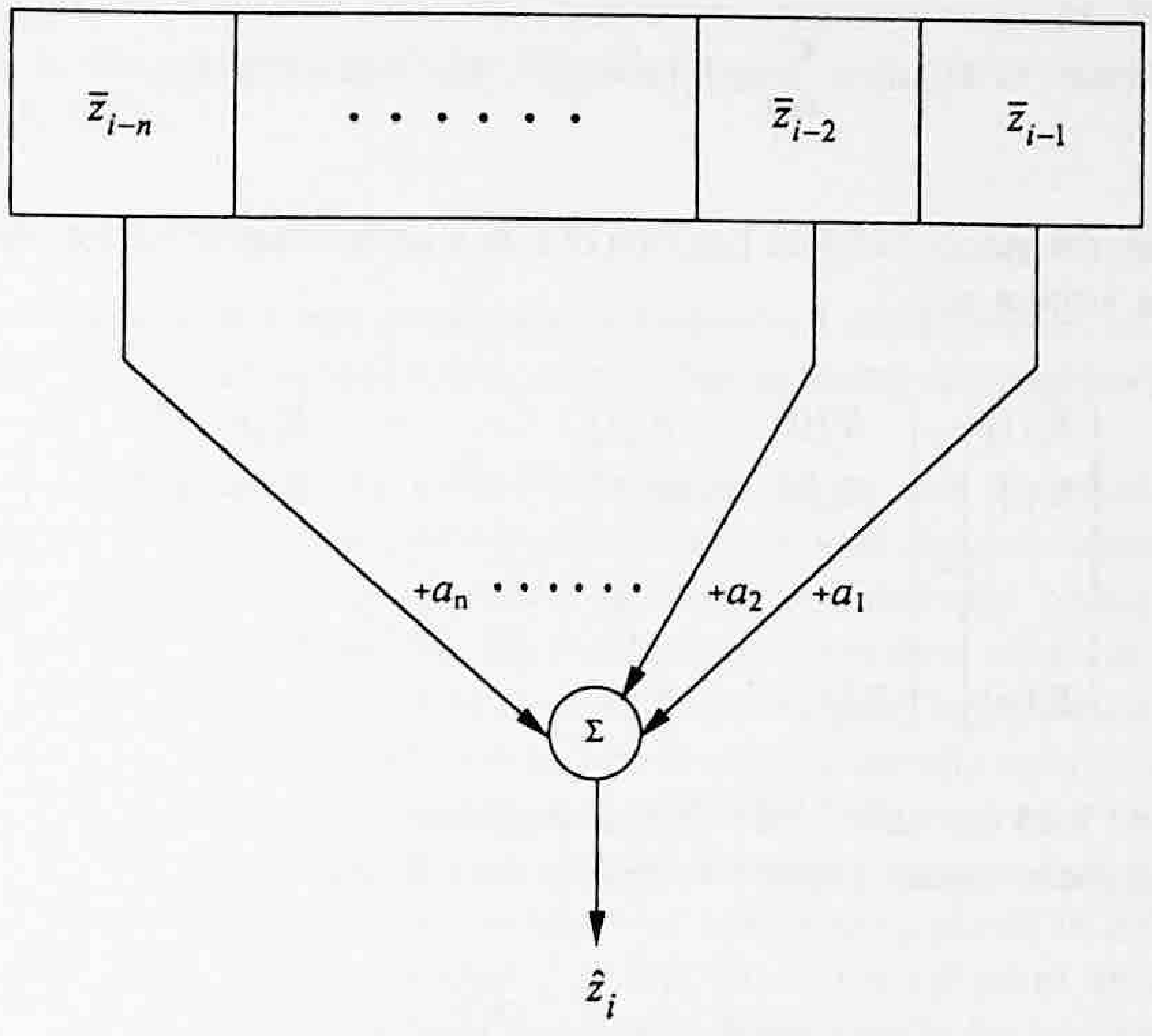


FIGURE 3.5 An illustration of a linear predictor.

The mean square prediction error, MSE_p, is

$$MSE_p = E\left[\left(e_p\right)^2\right] = E\left[\left(z_i - \hat{z}_i\right)^2\right] \tag{3.22}$$

The optimum prediction, then, refers to the determination of a set of coefficients a_j, $j = 1,2,\cdots,n$ such that the mean square prediction error, MSE_p, is minimized.

This optimization problem turns out to be computationally intractable for most practical cases due to the feedback around the quantizer shown in Figure 3.4, and the nonlinear nature of the quantizer. Therefore, the optimization problem is solved in two separate stages. That is, the best linear predictor is first designed ignoring the quantizer. Then, the quantizer is optimized for the distribution of the difference signal (Habibi, 1971). Although the predictor thus designed is suboptimal, ignoring the quantizer in the optimum predictor design allows us to substitute the reconstructed $\bar{z}_{i-j}$ by z_{i-j} for $j = 1,2,\cdots,n$, according to Equation 3.20. Consequently, we can apply the theory of optimum linear prediction to handle the design of the optimum predictor as shown below.

3.2.2 Orthogonality Condition and Minimum Mean Square Error

By taking the differentiation of MSE_p with respect to coefficient a_js, one can derive the following necessary conditions, which are usually referred to as the *orthogonality condition*:

$$E\left[e_p \cdot z_{i-j}\right] = 0 \quad for \quad j = 1,2,\cdots,n \tag{3.23}$$

The interpretation of Equation 3.23 is that the prediction error, e_p, must be orthogonal to all the observations, which are now the preceding samples: z_{i-j}, $j = 1,2,\cdots,n$ according to our discussion in Section 3.2.1. These are equivalent to

$$R_z(m) = \sum_{j=1}^{n} a_j R_z(m-j) \quad for \quad m = 1,2,\cdots,n \tag{3.24}$$

where R_z represents the autocorrelation function of z. In a vector-matrix format, the above orthogonal conditions can be written as

$$\begin{bmatrix} R_z(1) \\ R_z(2) \\ \vdots \\ \vdots \\ R_z(n) \end{bmatrix} = \begin{bmatrix} R_z(0) & R_z(1) & \cdots & \cdots & R_z(n-1) \\ R_z(1) & R_z(0) & \cdots & \cdots & R_z(n-2) \\ \vdots & \vdots & \cdots & \cdots & \vdots \\ \vdots & \vdots & \cdots & \cdots & \vdots \\ R_z(n-1) & R_z(n) & \cdots & \cdots & R_z(0) \end{bmatrix} \cdot \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ \vdots \\ a_n \end{bmatrix} \tag{3.25}$$

Equations 3.24 and 3.25 are called Yule-Walker equations.

The minimum mean square prediction error is then found to be

$$MSE_p = R_z(0) - \sum_{j=1}^{n} a_j R_z(j) \tag{3.26}$$

These results can be found in texts dealing with random processes, e.g., in (Leon-Garcia, 1994).

3.2.3 Solution to Yule-Walker Equations

Once autocorrelation data are available, the Yule-Walker equation can be solved by matrix inversion. A recursive procedure was developed by Levinson to solve the Yule-Walker equations (Leon-Garcia, 1993). When the number of previous samples used in the linear predictor is large, i.e., the dimension of the matrix is high, the Levinson recursive algorithm becomes more attractive. Note that in the field of image coding the autocorrelation function of various types of video frames is derived from measurements (O'Neal, 1966; Habibi, 1971).

3.3 SOME ISSUES IN THE IMPLEMENTATION OF DPCM

Several related issues in the implementation of DPCM are discussed in this section.

3.3.1 Optimum DPCM System

Since DPCM consists mainly of two parts, prediction and quantization, its optimization should not be carried out separately. The interaction between the two parts is quite complicated, however, and thus combined optimization of the whole DPCM system is difficult. Fortunately, with the mean square error criterion, the relation between quantization error and prediction error has been found:

$$MSE_q \approx \frac{9}{2N^2} MSE_p \tag{3.27}$$

where N is the total number of reconstruction levels in the quantizer (O'Neal, 1966; Musmann, 1979). That is, the mean square error of quantization is approximately proportional to the mean square error of prediction. With this approximation, we can optimize the two parts separately, as mentioned in Section 3.2.1. While the optimization of quantization was addressed in Chapter 2, the optimum predictor was discussed in Section 3.2. A large amount of work has been done on this subject. For instance, the optimum predictor for color image coding was designed and tested in (Pirsch and Stenger, 1977).

3.3.2 1-D, 2-D, AND 3-D DPCM

In Section 3.1.2, we expressed linear prediction in Equation 3.15. However, so far we have not really discussed how to predict a pixel's gray level value by using its neighboring pixels' coded gray level values.

Recall that a practical pixel-to-pixel differential coding system was discussed in Section 3.1.1. There, the reconstructed intensity of the immediately preceding pixel along the same scan line is used as a prediction of the pixel intensity being coded. This type of differential coding is referred to as 1-D DPCM. In general, 1-D DPCM may use the reconstructed gray level values of more than one of the preceding pixels within the same scan line to predict that of a pixel being coded. By far, however, the immediately preceding pixel in the same scan line is most frequently used in 1-D DPCM. That is, pixel A in Figure 3.6 is often used as a prediction of pixel Z, which is being DPCM coded.

Sometimes in DPCM image coding, both the decoded intensity values of adjacent pixels within the same scan line and the decoded intensity values of neighboring pixels in different scan lines are involved in the prediction. This is called 2-D DPCM. A typical pixel arrangement in 2-D predictive coding is shown in Figure 3.6. Note that the pixels involved in the prediction are restricted to be either in the lines above the line where the pixel being coded, Z, is located or on the left-hand side of pixel Z if they are in the same line. Traditionally, a TV frame is scanned from top to bottom and from left to right. Hence, the above restriction indicates that only those pixels, which have been coded and available in both the transmitter and the receiver, are used in the prediction. In 2-D system theory, this support is referred to as recursively computable (Bose, 1982). An often-used 2-D prediction involves pixels A, D, and E.

Obviously, 2-D predictive coding utilizes not only the spatial correlation existing within a scan line but also that existing in neighboring scan lines. In other words, the spatial correlation is utilized both horizontally and vertically. It was reported that 2-D predictive coding outperforms 1-D predictive coding by decreasing the prediction error by a factor of two, or equivalently, 3dB in *SNR*. The improvement in subjective assessment is even larger (Musmann, 1979). Furthermore, the transmission error in 2-D predictive image coding is much less severe than in 1-D predictive image coding. This is discussed in Section 3.6.

In the context of image sequences, neighboring pixels may be located not only in the same image frame but also in successive frames. That is, neighboring pixels along the time dimension are also involved. If the prediction of a DPCM system involves three types of neighboring pixels: those along the same scan line, those in the different scan lines of the same image frame, and those

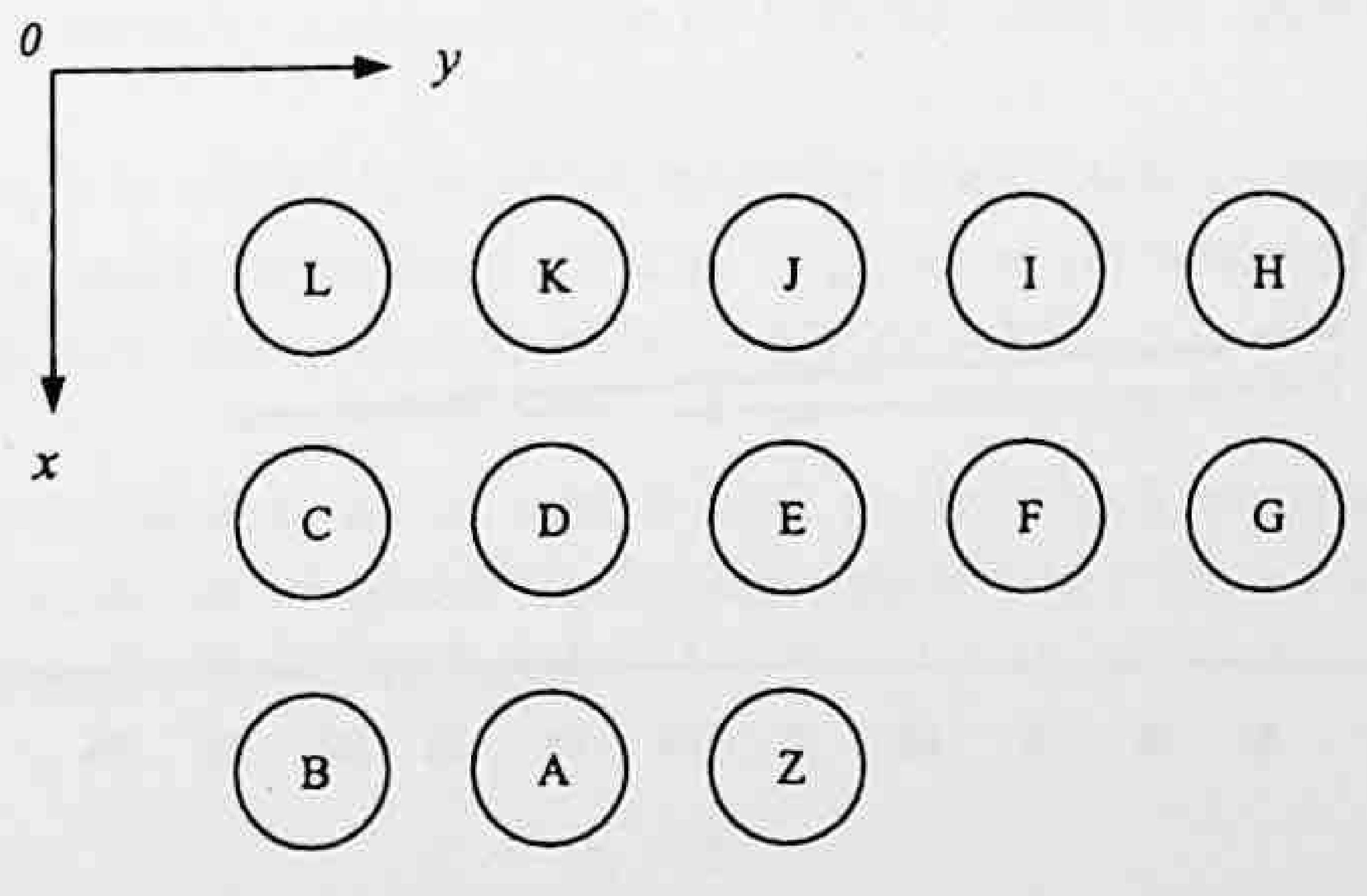


FIGURE 3.6 Pixel arrangement in 1-D and 2-D prediction.

in the different frames, the DPCM is then called 3-D differential coding. It will be discussed in Section 3.5.

3.3.3 Order of Predictor

The number of coefficients in the linear prediction, n, is referred to as the order of the predictor. The relation between the mean square prediction error, MSE_p, and the order of the predictor, n, has been studied. As shown in Figure 3.7, the MSE_p decreases quite effectively as n increases, but the performance improvement becomes negligible as $n > 3$ (Habibi, 1971).

3.3.4 Adaptive Prediction

Adaptive DPCM means adaptive prediction and adaptive quantization. As adaptive quantization was discussed in Chapter 2, here we discuss adaptive prediction only.

Similar to the discussion on adaptive quantization, adaptive prediction can be done in two different ways: forward adaptive and backward adaptive prediction. In the former, adaptation is based on the input of a DPCM system, while in the latter, adaptation is based on the output of the DPCM. Therefore, forward adaptive prediction is more sensitive to changes in local statistics. Prediction parameters (the coefficients of the predictor), however, need to be transmitted as side information to the decoder. On the other hand, quantization error is involved in backward adaptive prediction. Hence, the adaptation is less sensitive to local changing statistics. But, it does not need to transmit side information.

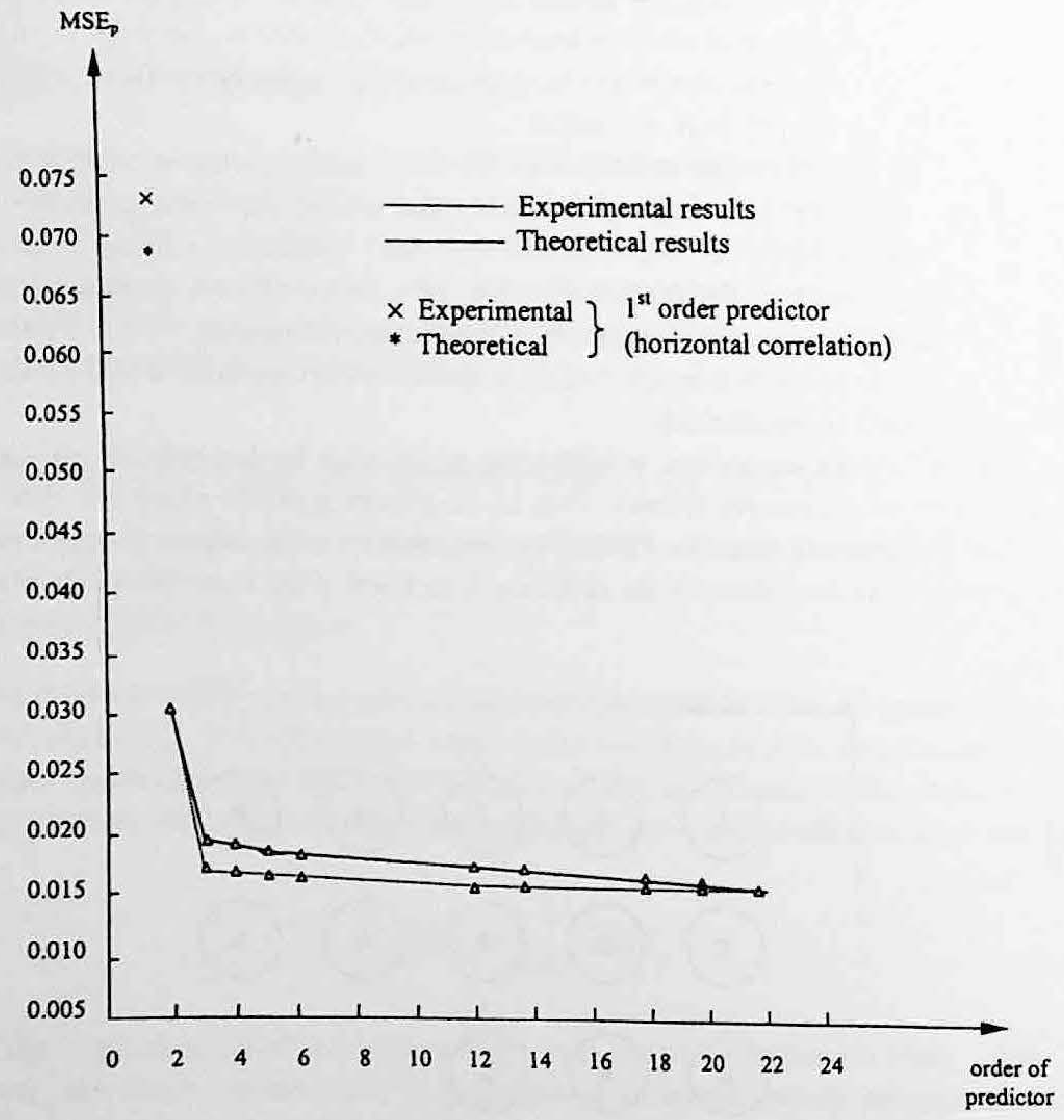


FIGURE 3.7 Mean square prediction error vs. order of predictor. (Data from Habibi, 1971.)

In either case, the data (either input or output) have to be buffered. Autocorrelation coefficients are analyzed, based on which prediction parameters are determined.

3.3.5 Effect of Transmission Errors

Transmission errors caused by channel noise may reverse the binary bit information from 0 to 1 or 1 to 0 with what is known as *bit error probability*, or *bit error rate*. The effect of transmission errors on reconstructed images varies depending on different coding techniques.

In the case of the PCM-coding technique, each pixel is coded independently of the others. Therefore bit reversal in the transmission only affects the gray level value of the corresponding pixel in the reconstructed image. It does not affect other pixels in the reconstructed image.

In DPCM, however, the effect caused by transmission errors becomes more severe. Consider a bit reversal occurring in transmission. It causes error in the corresponding pixel. But, this is not the end of the effect. The affected pixel causes errors in reconstructing those pixels towards which the erroneous gray level value was used in the prediction. In this way, the transmission error propagates.

Interestingly, it is reported that the error propagation is more severe in 1-D differential image coding than in 2-D differential coding. This may be explained as follows. In 1-D differential coding, usually only the immediate preceding pixel in the same scan line is involved in prediction. Therefore, an error will be propagated along the scan line until the beginning of the next line, where the pixel gray level value is reinitialized. In 2-D differential coding, the prediction of a pixel gray level value depends not only on the reconstructed gray level values of pixels along the same scan line but also on the reconstructed gray level values of the vertical neighbors. Hence, the effect caused by a bit reversal transmission error is less severe than in the 1-D differential coding.

For this reason, the bit error rate required by DPCM coding is lower than that required by PCM coding. For instance, while a bit error rate less than $5 \cdot 10^{-6}$ is normally required for PCM to provide broadcast TV quality, for the same application a bit error rate less than 10^{-7} and 10^{-9} are required for DPCM coding with 2-D prediction and 1-D prediction, respectively (Musmann, 1979).

Channel encoding with an error correction capability was applied to lower the bit error rate. For instance, to lower the bit error rate from the order of 10^{-6} to the order of 10^{-9} for DPCM coding with 1-D prediction, an error correction by adding 3% redundancy in channel coding has been used (Bruders, 1978).

3.4 DELTA MODULATION

Delta modulation (DM) is an important, simple, special case of DPCM, as discussed above. It has been widely applied and is thus an important coding technique in and of itself.

The above discussion and characterization of DPCM systems are applicable to DM systems. This is because DM is essentially a special type of DPCM, with the following two features.

1. The linear predictor is of the first order, with the coefficient a_1 equal to 1.
2. The quantizer is a one-bit quantizer. That is, depending on whether the difference signal is positive or negative, the output is either $+\Delta/2$ or $-\Delta/2$.

To perceive these two features, let us take a look at the block diagram of a DM system and the input-output characteristic of its one-bit quantizer, shown in Figures 3.8 and 3.9, respectively. Due to the first feature listed above, we have:

$$\hat{z}_i = \bar{z}_{i-1} \tag{3.28}$$

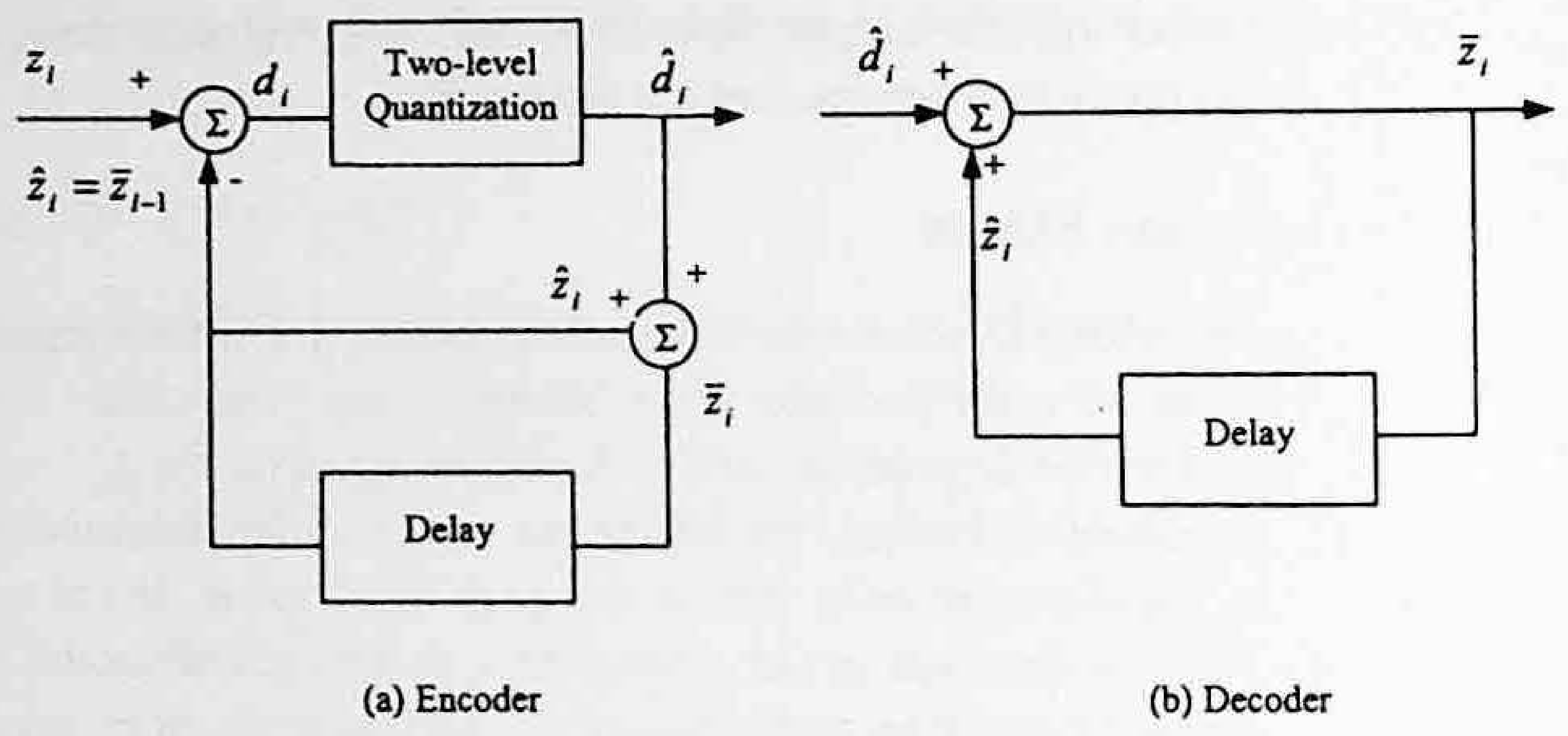


FIGURE 3.8 Block diagram of DM systems.

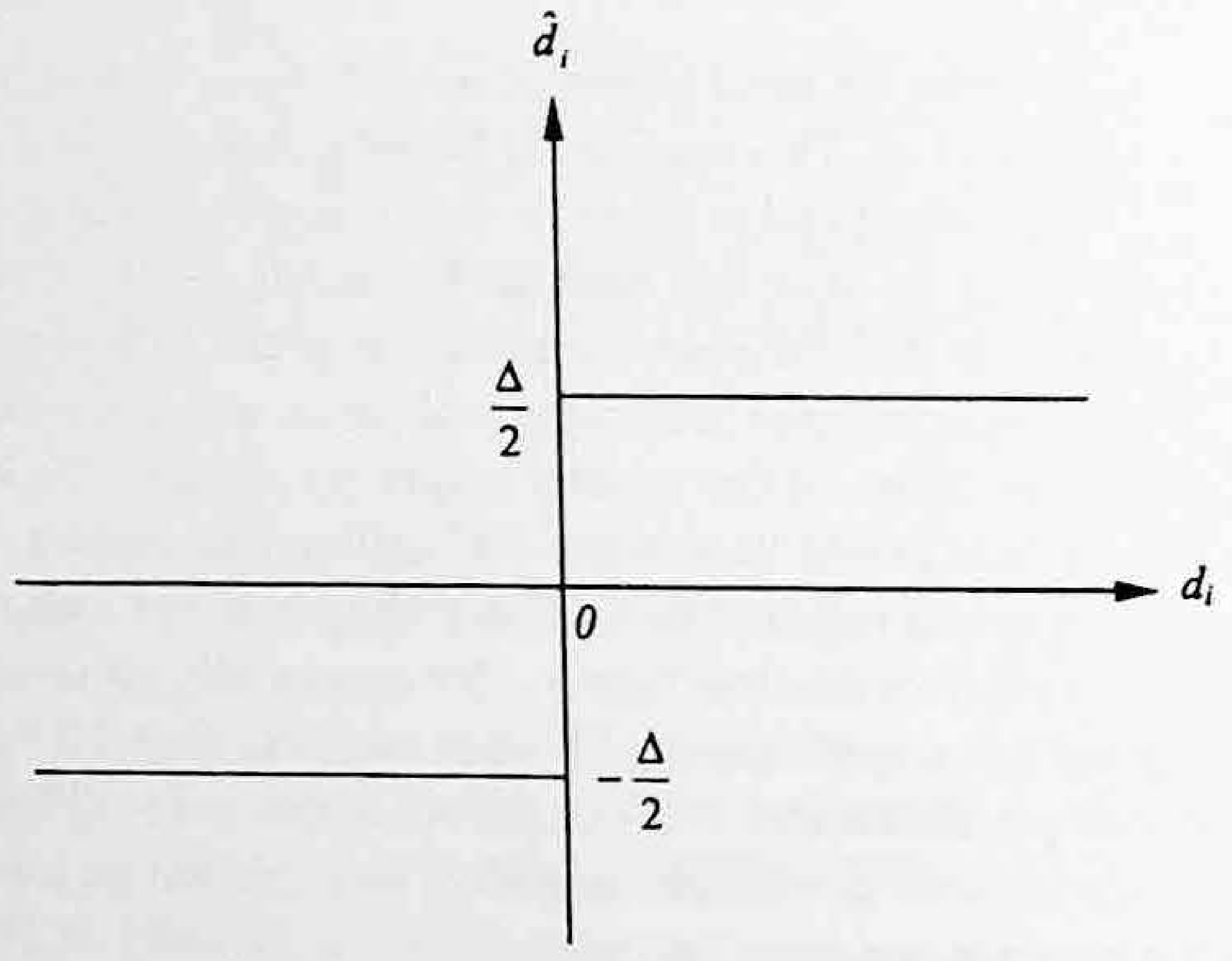


FIGURE 3.9 Input-output characteristic of two-level quantization in DM.

Next, we see that there are only two reconstruction levels in quantization because of the second feature. That is,

$$\hat{d}_i = \begin{cases} +\Delta/2 & if \quad z_i > \bar{z}_{i-1} \\ -\Delta/2 & if \quad z_i < \bar{z}_{i-1} \end{cases} \tag{3.29}$$

From the relation between the reconstructed value and the predicted value of DPCM, discussed above, and the fact that DM is a special case of DPCM, we have

$$\bar{z}_i = \hat{z}_i + \hat{d}_i \tag{3.30}$$

Combining Equations 3.28, 3.29, and 3.30, we have

$$\bar{z}_i = \begin{cases} \bar{z}_{i-1} + \Delta/2 & if \quad z_i > \bar{z}_{i-1} \\ \bar{z}_{i-1} - \Delta/2 & if \quad z_i < \bar{z}_{i-1} \end{cases} \tag{3.31}$$

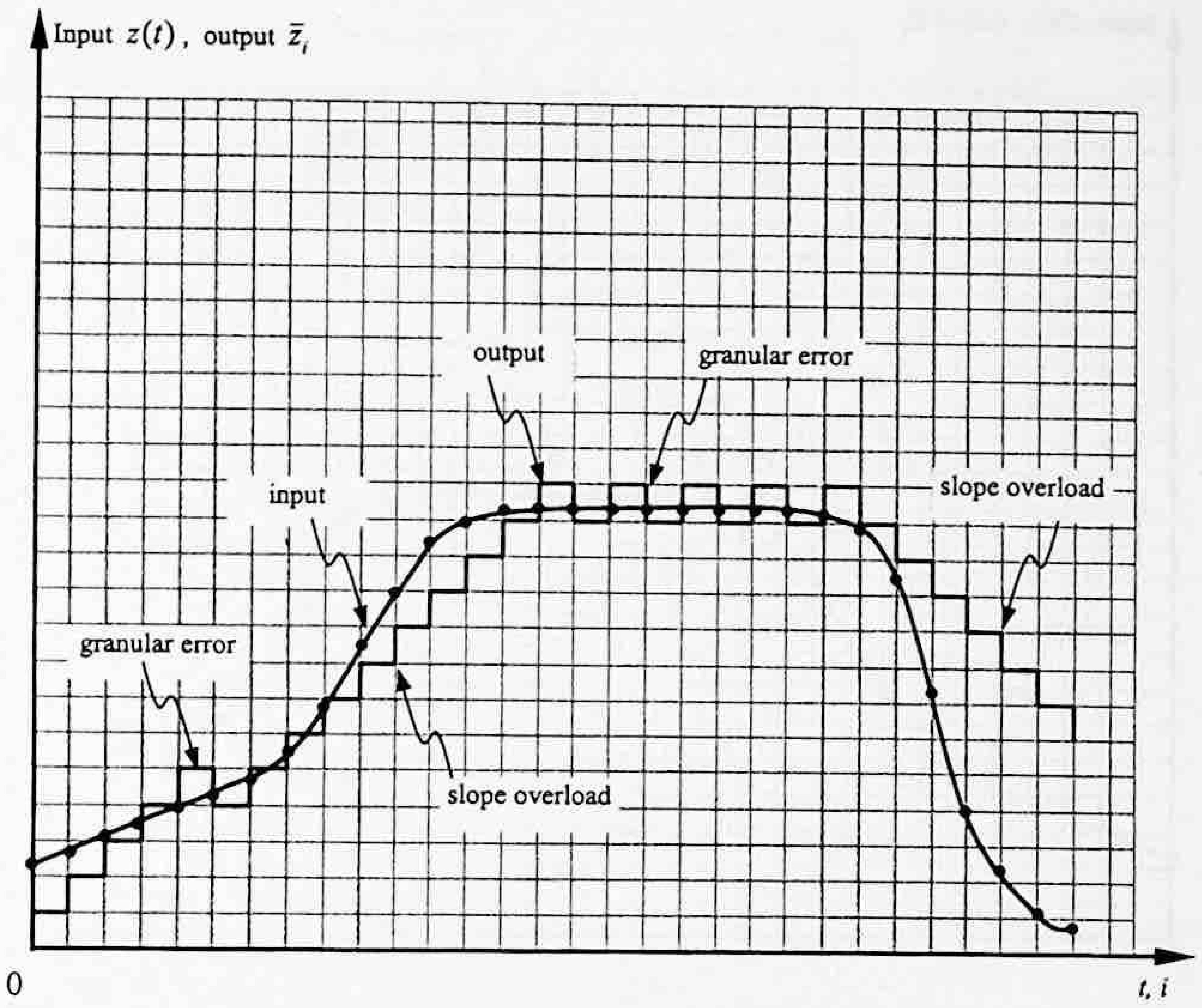


FIGURE 3.10 DM with fixed step size.

The above mathematical relationships are of importance in understanding DM systems. For instance, Equation 3.31 indicates that the step size Δ of DM is a crucial parameter. We notice that a large step size compared with the magnitude of the difference signal causes granular error, as shown in Figure 3.10. Therefore, in order to reduce the granular error we should choose a small step size. On the other hand, a small step size compared with the magnitude of the difference signal will lead to the overload error discussed in Chapter 2 for quantization. Since in DM systems it is the difference signal that is quantized, the overload error in DM becomes *slope overload* error, as shown in Figure 3.10. That is, it takes a while (multiple steps) for the reconstructed samples to catch up with the sudden change in input. Therefore, the step size should be large in order to avoid the slope overload. Considering these two conflicting factors, a proper compromise in choosing the step size is common practice in DM.

To improve the performance of DM, an oversampling technique is often applied. That is, the input is oversampled prior to the application of DM. By oversampling, we mean that the sampling frequency is higher than the sampling frequency used in obtaining the original input signal. The increased sample density caused by oversampling decreases the magnitude of the difference signal. Consequently, a relatively small step size can be used so as to decrease the granular noise without increasing the slope overload error. Thus, the resolution of the DM-coded image is kept the same as that of the original input (Jayant, 1984; Lim, 1990).

To achieve better performance for changing inputs, an adaptive technique can be applied in DM. That is, either input (forward adaptation) or output (backward adaptation) data are buffered and the data variation is analyzed. The step size is then chosen accordingly. If it is forward adaptation, side information is required for transmission to the decoder. Figure 3.11 demonstrates step size adaptation. We see the same input as that shown in Figure 3.10. But, the step size is now not fixed. Instead, the step size is adapted according to the varying input. When the input changes with a large slope, the step size increases to avoid the slope overload error. On the other hand, when the input changes slowly, the step size decreases to reduce the granular error.

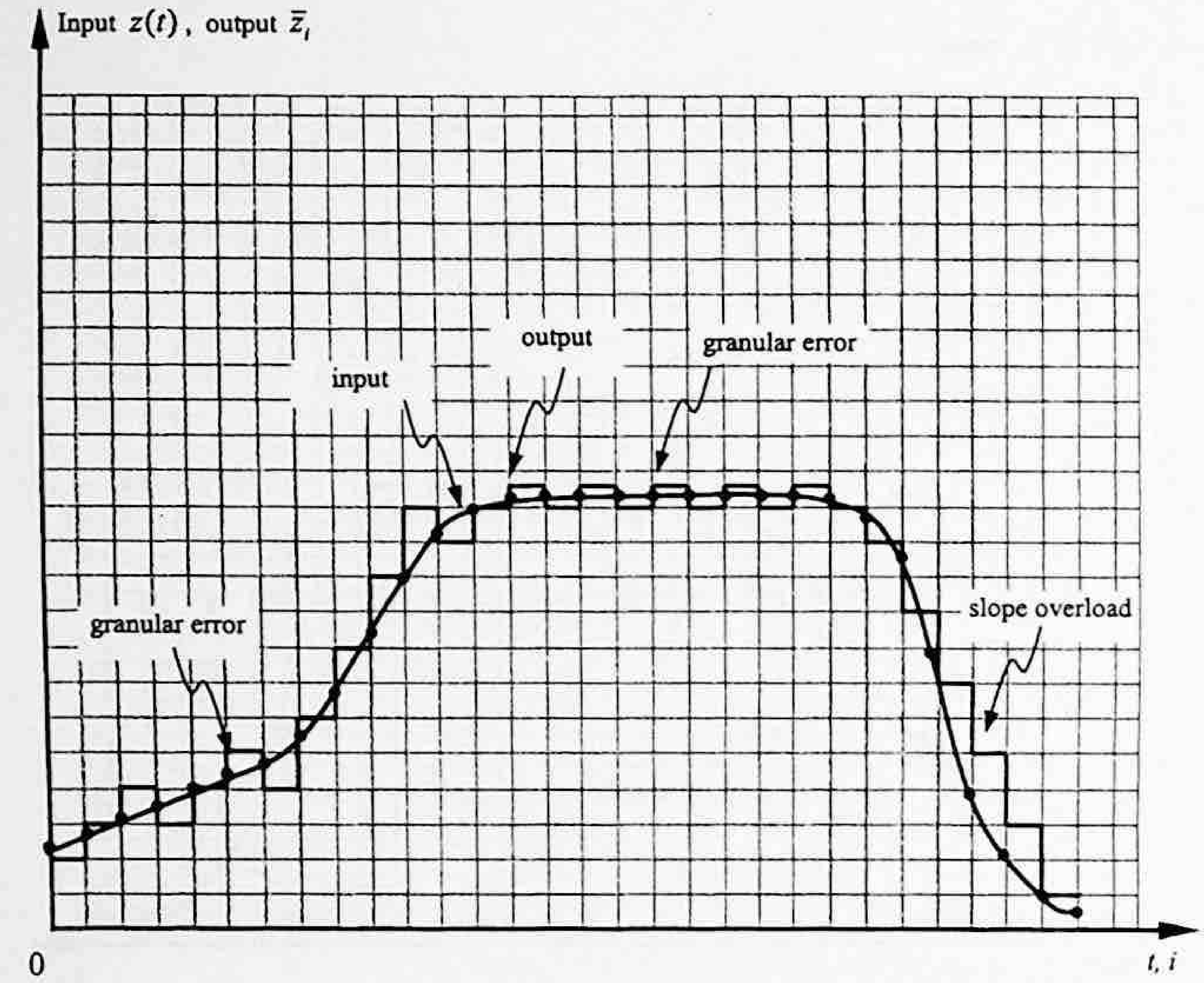


FIGURE 3.11 Adaptive DM.

3.5 INTERFRAME DIFFERENTIAL CODING

As was mentioned in Section 3.3.2, 3-D differential coding involves an image sequence. Consider a sensor located in 3-D world space. For instance, in applications such as videophony and videoconferencing, the sensor is fixed in position for a while and it takes pictures. As time goes by, the images form a temporal image sequence. The coding of such an image sequence is referred to as interframe coding. The subject of image sequence and video coding is addressed in Sections III and IV. In this section, we briefly discuss how differential coding is applied to interframe coding.

3.5.1 CONDITIONAL REPLENISHMENT

Recognizing the great similarity between consecutive TV frames, a conditional replenishment coding technique was proposed and developed (Mounts, 1969). It was regarded as one of the first real demonstrations of interframe coding exploiting interframe redundancy (Netravali and Robbins, 1979).

In this scheme, the previous frame is used as a reference for the present frame. Consider a pair of pixels: one in the previous frame, the other in the present frame — both occupying the same spatial position in the frames. If the gray level difference between the pair of pixels exceeds a certain criterion, then the pixel is considered a *changing* pixel. The present pixel gray level value and its position information are transmitted to the receiving side, where the pixel is replenished. Otherwise, the pixel is considered *unchanged*. At the receiver its previous gray level is repeated. A block diagram of conditional replenishment is shown in Figure 3.12. There, a frame memory unit in the transmitter is used to store frames. The differencing and thresholding of corresponding pixels in two consecutive frames can then be conducted there. A buffer in the transmitter is used to smooth the transmission data rate. This is necessary because the data rate varies from region to region within an image frame and from frame to frame within an image sequence. A buffer in the receiver is needed for a similar consideration. In the frame memory unit, the replenishment is

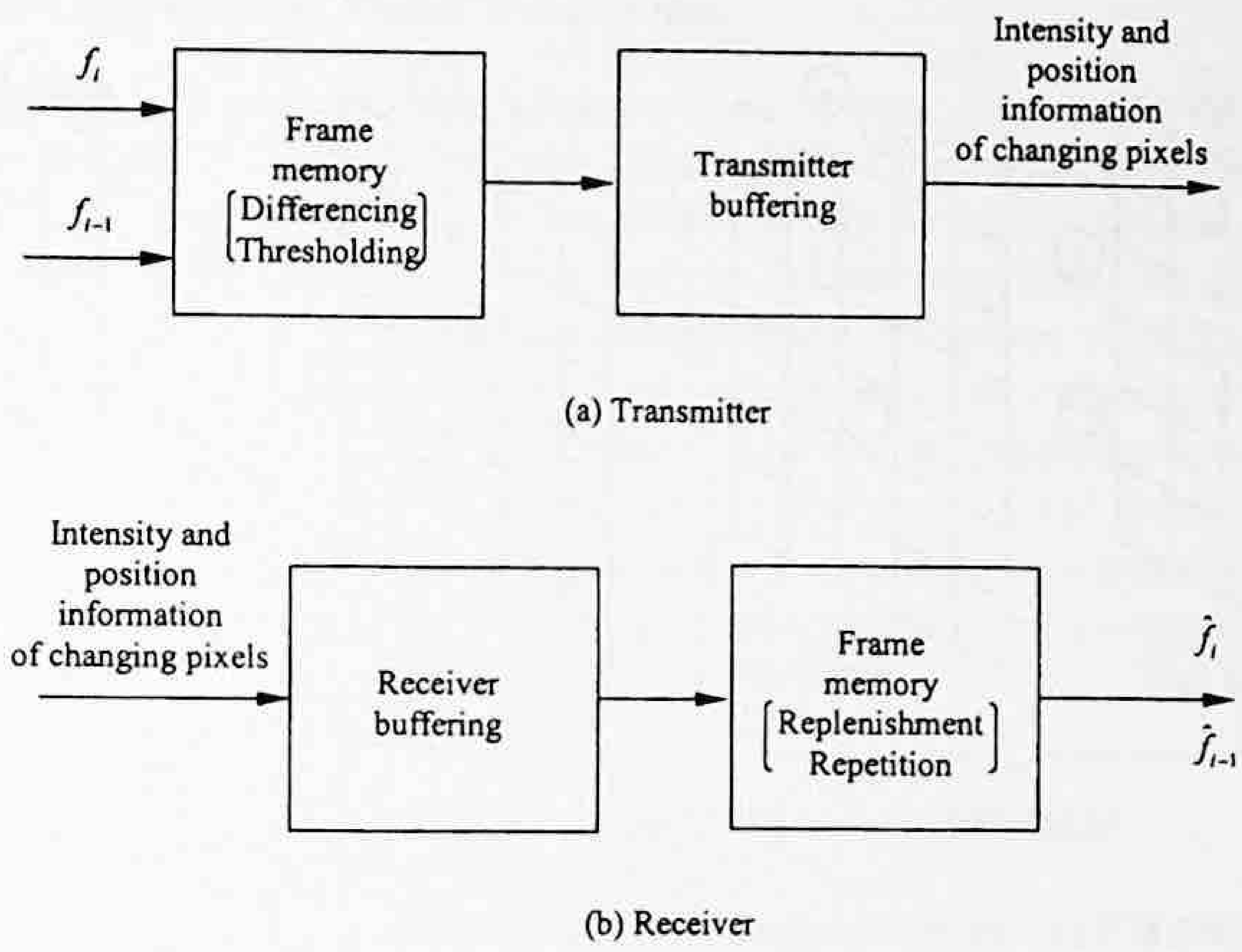


FIGURE 3.12 Block diagram of conditional replenishment.

carried out for the changing pixels and the gray level values in the receiver are repeated for the unchanged pixels.

With conditional replenishment, a considerable savings in bit rate was achieved in applications such as videophony, videoconferencing, and TV broadcasting. Experiments in real time, using the head-and-shoulder view of a person in animated conversation as the video source, demonstrated an average bit rate of 1 bit/pixel with a quality of reconstructed video comparable with standard 8 bit/pixel PCM transmission (Mounts, 1969). Compared with pixel-to-pixel 1-D DPCM, the most popularly used coding technique at the time, the conditional replenishment technique is more efficient due to the exploitation of high interframe redundancy. As pointed in (Mounts, 1969), there is more correlation between television pixels along the frame-to-frame temporal dimension than there is between adjacent pixels within a signal frame. That is, the temporal redundancy is normally higher than spatial redundancy for TV signals.

Tremendous efforts have been made to improve the efficiency of this rudimentary technique. For an excellent review, readers are referred to (Haskell et al., 1972, 1979). 3-D DPCM coding is among the improvements and is discussed next.

3.5.2 3-D DPCM

It was soon realized that it is more efficient to transmit the gray level difference than to transmit the gray level itself, resulting in interframe differential coding. Furthermore, instead of treating each pixel independently of its neighboring pixels, it is more efficient to utilize spatial redundancy as well as temporal redundancy, resulting in 3-D DPCM.

Consider two consecutive TV frames, each consisting of an odd and an even field. Figure 3.13 demonstrates the small neighborhood of a pixel, Z, in the context. As with the 1-D and 2-D DPCM discussed before, the prediction can only be based on the previously encoded pixels. If the pixel under consideration, Z, is located in the even field of the present frame, then the odd field of the present frame and both odd and even fields of the previous frame are available. As mentioned in Section 3.3.2, it is assumed that in the even field of the present frame, only those pixels in the lines above the line where pixel Z lies and those pixels left of the Z in the line where Z lies are used for prediction.

Table 3.1 lists several utilized linear prediction schemes. It is recognized that the case of *element difference* is a 1-D predictor since the immediately preceding pixel is used as the predictor. The

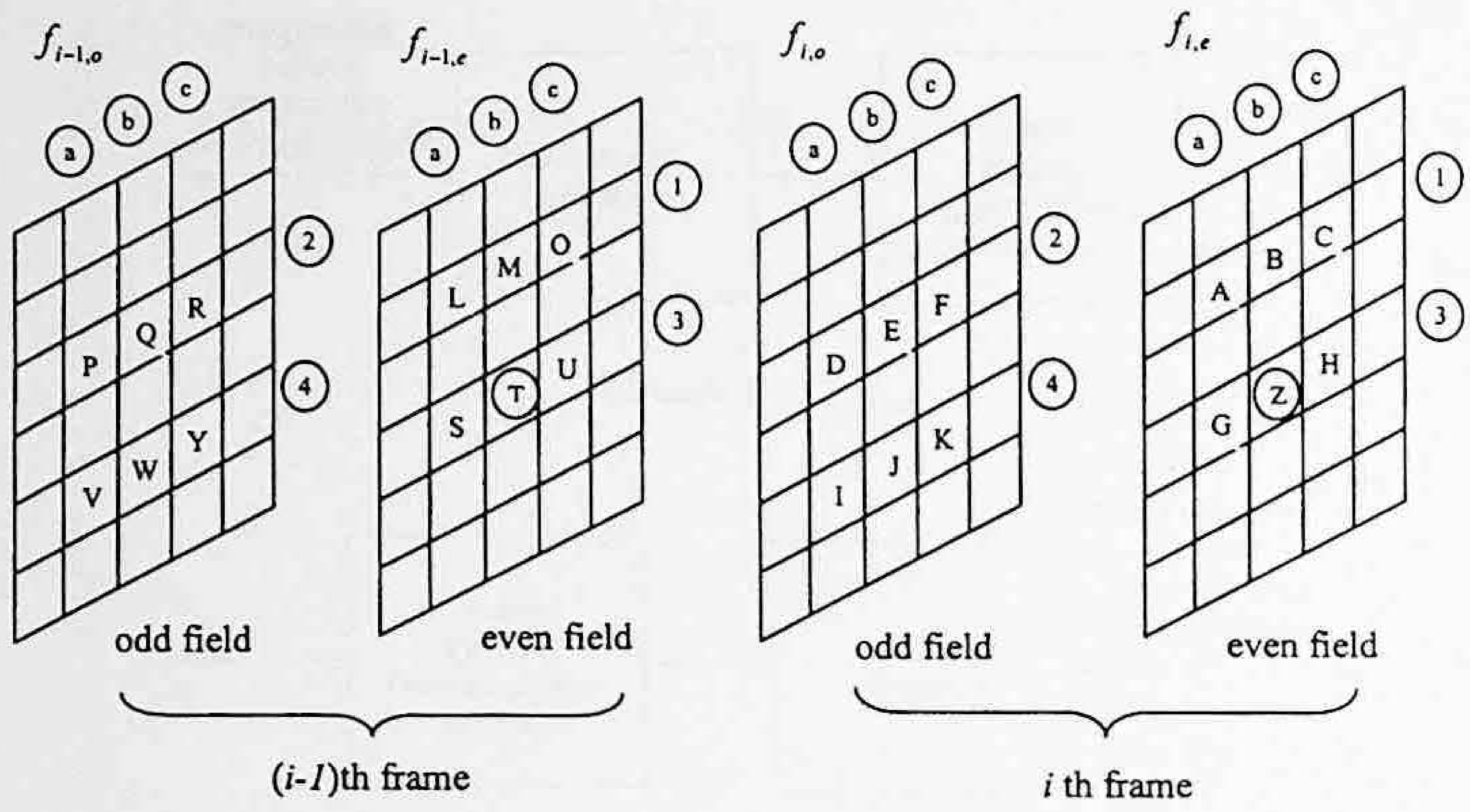


FIGURE 3.13 Pixel arrangement in two TV frames. (After Haskell, 1979.)

field difference is defined as the arithmetic average of two immediately vertical neighboring pixels in the previous odd field. Since the odd field is generated first, followed by the even field, this predictor cannot be regarded as a pure 2-D predictor. Instead, it should be considered a 3-D predictor. The remaining cases are all 3-D predictors. One thing is common in all the cases: the gray levels of pixels used in the prediction have already been coded and thus are available in both the transmitter and the receiver. The prediction error of each changing pixel Z identified in thresholding process is then quantized and coded.

An analysis of the relationship between the entropy of moving areas (bits per changing pixel) and the speed of the motion (pixels per frame interval) in an image containing a moving mannequin's head was studied with different linear predictions, as listed in Table 3.1 in Haskell (1979). It was found that the element difference of field difference generally corresponds to the lowest entropy, meaning that this prediction is the most efficient. The frame difference and element difference correspond to higher entropy. It is recognized that, in the circumstances, transmission error will be propagated if the pixels in the previous line are used in prediction (Connor, 1973). Hence, the linear predictor should use only pixels from the same line or the same line in the previous frame when bit reversal error in transmission needs to be considered. Combining these two factors, the element difference of frame difference prediction is preferred.

TABLE 3.1
Some Linear Prediction Schemes. (After Haskell, 1979).

	Original signal (Z)	Prediction signal ($\hat{Z}$)	Differential signal (d_z)
Element difference	Z	G	$Z-G$
Field difference	Z	$\frac{E+J}{2}$	$Z-\frac{E+J}{2}$
Frame difference	Z	T	$Z-T$
Element difference of frame difference	Z	$T+G-S$	$(Z-G)-(T-S)$
Line difference of frame difference	Z	$T+B-M$	$(Z-B)-(T-M)$
Element difference of field difference	Z	$T+\frac{E+J}{2}-\frac{Q+W}{2}$	$\left(Z-\frac{E+J}{2}\right)-\left(T-\frac{Q+W}{2}\right)$

3.5.3 Motion-Compensated Predictive Coding

When frames are taken densely enough, changes in successive frames can be attributed to the motion of objects during the interval between frames. Under this assumption, if we can analyze object motion from successive frames, then we should be able to predict objects in the next frame based on their positions in the previous frame and the estimated motion. The difference between the original frame and the predicted frame thus generated and the motion vectors are then quantized and coded. If the motion estimation is accurate enough, the motion-compensated prediction error can be smaller than 3-D DPCM. In other words, the variance of the prediction error will be smaller, resulting in more efficient coding. Take motion into consideration — this differential technique is called motion compensated predictive coding. This has been a major development in image sequence coding since the 1980s. It has been adopted by all international video coding standards. A more detailed discussion is provided in Chapter 10.

3.6 INFORMATION-PRESERVING DIFFERENTIAL CODING

As emphasized in Chapter 2, quantization is not reversible in the sense that it causes permanent information loss. The DPCM technique, discussed above, includes quantization, and hence is lossy coding. In applications such as those involving scientific measurements, information preservation is required. In this section, the following question is addressed: under these circumstances, how should we apply differential coding in order to reduce the bit rate while preserving information?

Figure 3.14 shows a block diagram of information-preserving differential coding. First, we see that there is no quantizer. Therefore, the irreversible information loss associated with quantization does not exist in this technique. Second, we observe that prediction and differencing are still used. That is, the differential (predictive) technique still applies. Hence it is expected that the variance of the difference signal is smaller than that of the original signal, as explained in Section 3.1. Consequently, the higher-peaked histograms make coding more efficient. Third, an efficient lossless coder is utilized. Since quantizers cannot be used here, PCM with natural binary coding is not used here. Since the histogram of the difference signal is narrowly concentrated about its mean, lossless coding techniques such as an efficient Huffman coder (discussed in Chapter 5) is naturally a suitable choice here.

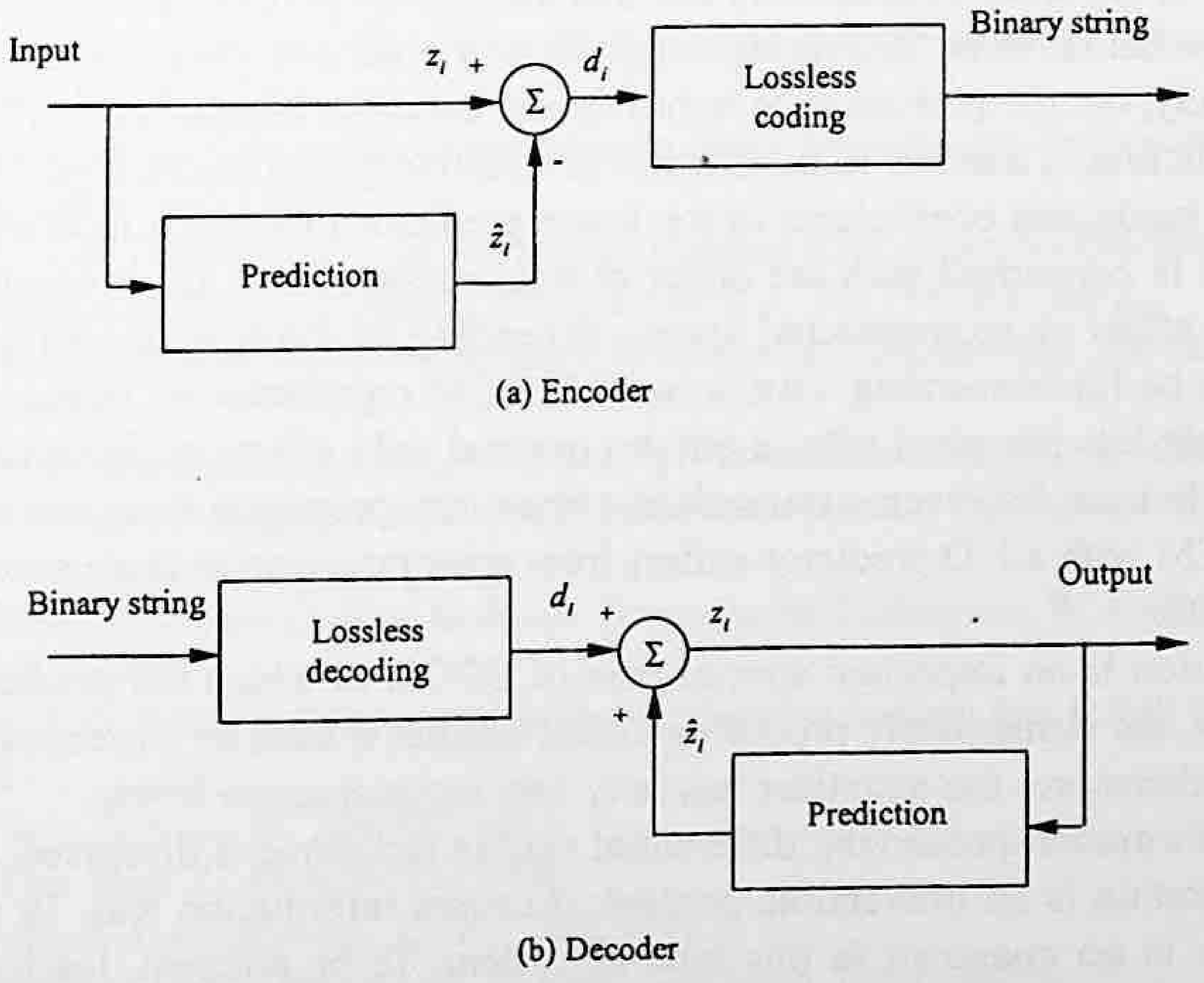


FIGURE 3.14 Block diagram of information-preserving differential coding.

As mentioned before, input images are normally in a PCM coded format with a bit rate of eight bits per pixel for monochrome pictures. The difference signal is therefore integer-valued. Having no quantization and using an efficient lossless coder, the coding system depicted in Figure 3.14, therefore, is an information-preserving differential coding technique.

3.7 SUMMARY

Rather than coding the signal itself, differential coding, also known as predictive coding, encodes the difference between the signal and its prediction. Utilizing spatial and/or temporal correlation between pixels in the prediction, the variance of the difference signal can be much smaller than that of the original signal, thus making differential coding quite efficient.

Among differential coding methods, differential pulse code modulation (DPCM) is used most widely. In DPCM coding, the difference signal is quantized and codewords are assigned to the quantized difference. Prediction and quantization are therefore two major components in the DPCM systems. Since quantization was addressed in Chapter 2, this chapter emphasizes prediction. The theory of optimum linear prediction is introduced. Here, optimum means minimization of the mean square prediction error. The formulation of optimum linear prediction, the orthogonality condition, and the minimum mean square prediction error are presented. The orthogonality condition states that the prediction error must be orthogonal to each observation, i.e., to the reconstructed sample intensity values used in the linear prediction. By solving the Yule-Walker equation, the optimum prediction coefficients may be determined.

In addition, some fundamental issues in implementing the DPCM technique are discussed. One issue is the dimensionality of the predictor in DPCM. We discussed 1-D, 2-D, and 3-D predictors. DPCM with a 2-D predictor demonstrates better performance than a 1-D predictor since 2-D DPCM utilizes more spatial correlation, i.e., not only horizontally but also vertically. As a result, a 3-dB improvement in *SNR* was reported. 3-D prediction is encountered in what is known as interframe coding. There, temporal correlation exists. 3-D DPCM utilizes both spatial and temporal correlation between neighboring pixels in successive frames. Consequently, more redundancy can be removed. Motion-compensated predictive coding is a very powerful technique in video coding related to differential coding. It uses a more advanced translational motion model in the prediction, however, and it is covered in Sections III and IV.

Another issue is the order of predictors and its effect on the performance of prediction in terms of mean square prediction error. Increasing the prediction order can lower the mean square prediction error effectively, but the performance improvement becomes insignificant after the third order.

Adaptive prediction is another issue. Similar to adaptive quantization, discussed in Chapter 2, we can adapt the prediction coefficients in the linear predictor to varying local statistics.

The last issue is concerned with the effect of transmission error. Bit reversal in transmission causes a different effect on reconstructed images depending on the type of coding technique used. PCM is known to be bit-consuming. (An acceptable PCM representation of monochrome images requires six to eight bits per pixel.) But a one-bit reversal only affects an individual pixel. For the DPCM coding technique, however, a transmission error may propagate from one pixel to the other. In particular, DPCM with a 1-D predictor suffers from error propagation more severely than DPCM with a 2-D predictor.

Delta modulation is an important special case of DPCM in which the predictor is of the first order. Specifically, the immediately preceding coded sample is used as a prediction of the present input sample. Furthermore, the quantizer has only two reconstruction levels.

Finally, an information-preserving differential coding technique is discussed. As mentioned in Chapter 2, quantization is an irreversible process: it causes information loss. In order to preserve information, there is no quantizer in this type of system. To be efficient, lossless codes such as Huffman code or arithmetic code should be used for difference signal encoding.

3.8 EXERCISES

3-1. Justify the necessity of the closed-loop DPCM with feedback around quantizers. That is, convince yourself why the quantization error will be accumulated if, instead of using the reconstructed preceding samples, we use the immediately preceding sample as the prediction of the sample being coded in DPCM.

3-2. Why does the overload error encountered in quantization appear to be the slope overload in DM?

3-3. What advantage does oversampling bring up in the DM technique?

3-4. What are the two features of DM that make it a subclass of DPCM?

3-5. Explain why DPCM with a 1-D predictor suffers from bit reversal transmission error more severely than DPCM with a 2-D predictor.

3-6. Explain why no quantizer can be used in information-preserving differential coding, and why the differential system can work without a quantizer.

3-7. Why do all the pixels involved in prediction of differential coding have to be in a recursively computable order from the point of view of the pixel being coded?

3-8. Discuss the similarity and dissimilarity between DPCM and motion compensated predictive coding.

REFERENCES

Bose, N. K. *Applied Multidimensional System Theory*, Van Nostrand Reinhold, New York, 1982.

Bruders, R., T. Kummerow, P. Neuhold, and P. Stamnitz, Ein versuchssystem zur digitalen ubertragung von fernsehsignalen unter besonderer berucksichtigung von ubertragungsfehlern, Festschrift 50 Jahre Heinrich-Hertz-Institut, Berlin, 1978.

Connor, D. J. *IEEE Trans. Commun.*, com-21, 695-706, 1973.

Cutler, C. C. U. S. Patent 2,605,361, 1952.

DeJager, F. *Philips Res. Rep.*, 7, 442-466, 1952.

Elias, P. *IRE Trans. Inf. Theory*, it-1, 16-32, 1955.

Habibi, A. Comparison of nth-order DPCM encoder with linear transformations and block quantization techniques, *IEEE Trans. Commun. Technol.*, COM-19(6), 948-956, 1971.

Harrison, C. W. *Bell Syst. Tech. J.*, 31, 764-783, 1952.

Haskell, B. G., F. W. Mounts, and J. C. Candy, Interframe coding of videotelephone pictures, *Proc. IEEE*, 60, 7, 792-800, 1972.

Haskell, B. G. Frame replenishment coding of television, in *Image Transmission Techniques*, W. K. Pratt (Ed.), Academic Press, New York, 1979.

Jayant, N. S. and P. Noll, *Digital Coding of Waveforms*, Prentice-Hall, Upper Saddle River, NJ, 1984.

Kretzmer, E. R. Statistics of television signals, *Bell Syst. Tech. J.*, 31, 751-763, 1952.

Leon-Garcia, A. *Probability and Random Processes for Electrical Engineering*, 2nd ed., Addison-Wesley, Reading, MA, 1994.

Lim, J. S. *Two-Dimensional Signal and Image Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1990.

Mounts, F. W. A video encoding system with conditional picture-element replenishment, *Bell Syst. Tech. J.*, 48, 7, 1969.

Musmann, H. G. Predictive Image Coding, in *Image Transmission Techniques*, W. K. Pratt (Ed.), Academic Press, New York, 1979.

Netravali, A. N. and J. D. Robbins, Motion-compensated television coding. Part I, *Bell Syst. Tech. J.*, 58, 3, 631-670, 1979.

Oliver, B. M. *Bell Syst. Tech. J.*, 31, 724-750, 1952.

O'Neal, J. B. *Bell Syst. Tech. J.*, 45, 689-721, 1966.

Pirsch, P. and L. Stenger, *Acta Electron.*, 19, 277-287, 1977.

Sayood, K. *Introduction to Data Compression*, Morgan Kaufmann, San Francisco, CA, 1996.

4 Transform Coding

As introduced in the previous chapter, differential coding achieves high coding efficiency by utilizing the correlation between pixels existing in image frames. Transform coding (TC), the focus of this chapter, is another efficient coding scheme based on utilization of interpixel correlation. As we will see in Chapter 7, TC has become a fundamental technique recommended by the international still image coding standard, JPEG. Moreover, TC has been found to be efficient in coding prediction error in motion-compensated predictive coding. As a result, TC was also adopted by the international video coding standards such as H.261, H.263, and MPEG 1, 2, and 4. This will be discussed in Section IV.

4.1 INTRODUCTION

Recall the block diagram of source encoders shown in Figure 2.3. There are three components in a source encoder: transformation, quantization, and codeword assignment. It is the transformation component that decides which format of input source is quantized and encoded. In DPCM, for instance, the difference between an original signal and a predicted version of the original signal is quantized and encoded. As long as the prediction error is small enough, i.e., the prediction resembles the original signal well (by using correlation between pixels), differential coding is efficient.

In transform coding, the main idea is that if the transformed version of a signal is less correlated compared with the original signal, then quantizing and encoding the transformed signal may lead to data compression. At the receiver, the encoded data are decoded and transformed back to reconstruct the signal. Therefore, in transform coding, the transformation component illustrated in Figure 2.3 is a transform. Quantization and codeword assignment are carried out with respect to the transformed signal, or, in other words, carried out in the transform domain.

We begin with the Hotelling transform, using it as an example of how a transform may decorrelate a signal in the transform domain.

4.1.1 HOTELLING TRANSFORM

Consider an N-dimensional vector $\vec{z}_s$. The ensemble of such vectors, $\{\vec{z}_s\}$ $s \in I$, where I represents the set of all vector indexes, can be modeled by a random vector $\bar{z}$ with each of its component z_i $i = 1, 2, \cdots, N$ as a random variable. That is,

$$\bar{z} = \left(z_1, z_2, \cdots, z_N\right)^T \tag{4.1}$$

where T stands for the operator of matrix transposition. The mean vector of the population, $m_{\bar{z}}$, is defined as

$$m_{\bar{z}} = E[\bar{z}] = \left(m_1, m_2, \cdots, m_N\right)^T \tag{4.2}$$

where E stands for the expectation operator. Note that $m_{\bar{z}}$ is an N-dimensional vector with the ith component, m_i, being the expectation value of the ith random variable component in $\bar{z}$.

$$m_i = E\left[z_i\right] \quad i = 1, 2, \cdots, N \tag{4.3}$$

The covariance matrix of the population, denoted by $C_{\vec{z}}$, is equal to

$$C_{\vec{z}} = E\left[\left(\vec{z} - m_{\vec{z}}\right)\left(\vec{z} - m_{\vec{z}}\right)^T\right]. \tag{4.4}$$

Note that the product inside the E operator is referred to as the *outer product* of the vector ($\vec{z} - m_{\vec{z}}$). Denote an entry at the ith row and jth column in the covariance matrix by $c_{i,j}$. From Equation 4.4, it can be seen that $c_{i,j}$ is the covariance between the ith and jth components of the random vector $\vec{z}$. That is,

$$c_{i,j} = E\left[\left(z_i - m_i\right)\left(z_j - m_j\right)\right] = Cov\left(z_i, z_j\right). \tag{4.5}$$

On the main diagonal of the covariance matrix $C_{\vec{z}}$, the element $c_{i,i}$ is the variance of the ith component of $\vec{z}$, z_i. Obviously, the covariance matrix $C_{\vec{z}}$ is a real and symmetric matrix. It is real because of the definition of random variables. It is symmetric because $Cov(z_i, z_j) = Cov(z_j, z_i)$. According to the theory of linear algebra, it is always possible to find a set of N orthonormal eigenvectors of the matrix $C_{\vec{z}}$, with which we can convert the real symmetric matrix $C_{\vec{z}}$ into a fully ranked diagonal matrix. This statement can be found in texts of linear algebra, e.g., in (Strang, 1998).

Denote the set of N orthonormal eigenvectors and their corresponding eigenvalues of the covariance matrix $C_{\vec{z}}$ by $\vec{e}_i$ and λ_i, $i = 1,2,\cdots,N$, respectively. Note that eigenvectors are column vectors. Form a matrix Φ such that its rows comprise the N transposed eigenvectors. That is,

$$\Phi = \left(\vec{e}_1, \vec{e}_2, \cdots, \vec{e}_N\right)^T. \tag{4.6}$$

Now, consider the following transformation:

$$\vec{y} = \Phi\left(\vec{z} - m_{\vec{z}}\right) \tag{4.7}$$

It is easy to verify that the transformed random vector $\vec{y}$ has the following two characteristics:

1. The mean vector, $m_{\vec{y}}$, is a zero vector. That is,

$$m_{\vec{y}} = 0. \tag{4.8}$$

2. The covariance matrix of the transformed random vector $C_{\vec{y}}$ is

$$C_{\vec{y}} = \Phi C_{\vec{z}} \Phi^T = \begin{bmatrix} \lambda_1 & & & & 0 \\ & \lambda_2 & & & \\ & & \ddots & & \\ & & & \ddots & \\ 0 & & & & \lambda_n \end{bmatrix}. \tag{4.9}$$

This transform is called the Hotelling transform (Hotelling, 1933), or eigenvector transform (Tasto, 1971; Wintz, 1972).

The inverse Hotelling transform is defined as

$$\vec{z} = \Phi^{-1}\vec{y} + m_{\vec{z}}, \tag{4.10}$$

where Φ^{-1} is the inverse matrix of Φ. It is easy to see from its formation discussed above that the matrix Φ is orthogonal. Therefore, we have $\Phi^T = \Phi^{-1}$. Hence, the inverse Hotelling transform can be expressed as

$$\vec{z} = \Phi^T \vec{y} + m_{\vec{z}} . \tag{4.11}$$

Note that in implementing the Hotelling transform, the mean vector $m_{\vec{z}}$ and the covariance matrix $C_{\vec{z}}$ can be calculated approximately by using a given set of K sample vectors (Gonzalez and Woods, 1992).

$$m_{\vec{z}} = \frac{1}{K}\sum_{s=1}^{K} \vec{z}_s \tag{4.12}$$

$$C_{\vec{z}} = \frac{1}{K}\sum_{s=1}^{K} \vec{z}_s \vec{z}_s^{\,T} - m_{\vec{z}} m_{\vec{z}}^T \tag{4.13}$$

The analogous transform for continuous data was devised by Karhunen and Loeve (Karhunen, 1947; Loeve, 1948). Alternatively, the Hotelling transform can be viewed as the discrete version of the Karhunen-Loeve transform (KLT). We observe that the covariance matrix $C_{\vec{y}}$ is a diagonal matrix. The elements in the diagonal are the eigenvalues of the covariance matrix $C_{\vec{z}}$. That is, the two covariance matrices have the same eigenvalues and eigenvectors because the two matrices are similar. The fact that zero values are everywhere except along the main diagonal in $C_{\vec{y}}$ indicates that the components of the transformed vector $\vec{y}$ are uncorrelated. That is, the correlation previously existing between the different components of the random vector $\vec{z}$ has been removed in the transformed domain. Therefore, if the input is split into *blocks* and the Hotelling transform is applied blockwise, the coding may be more efficient since the data in the transformed block are uncorrelated. At the receiver, we may produce a replica of the input with an inverse transform. This basic idea behind transform coding will be further illustrated next. Note that transform coding is also referred to as block quantization (Huang, 1963).

4.1.2 Statistical Interpretation

Let's continue our discussion of the 1-D Hotelling transform. Recall that the covariance matrix of the transformed vector $\vec{y}$, $C_{\vec{y}}$, is a diagonal matrix. The elements in the main diagonal are eigenvalues of the covariance matrix $C_{\vec{y}}$. According to the definition of a covariance matrix, these elements are the variances of the components of vector $\vec{y}$, denoted by $\sigma^2_{y,1}, \sigma^2_{y,2}, \cdots, \sigma^2_{y,N}$. Let us arrange the eigenvalues (variances) in a nonincreasing order. That is, $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N$. Choose an integer L, and $L < N$. Using the corresponding L eigenvectors, $\vec{e}_1, \vec{e}_2, \cdots, \vec{e}_L$, we form a matrix $\overline{\Phi}$ with these L eigenvectors (transposed) as its L rows. Obviously, the matrix $\overline{\Phi}$ is of $L \times N$. Hence, using the matrix $\overline{\Phi}$ in Equation 4.7 we will have the transformed vector $\vec{y}$ of $L \times 1$. That is,

$$\vec{y} = \overline{\Phi}\left(\vec{z} - m_{\vec{z}}\right). \tag{4.14}$$

The inverse transform changes accordingly:

$$\vec{z}' = \overline{\Phi}^T \vec{y} + m_{\vec{z}} . \tag{4.15}$$

Note that the reconstructed vector $\vec{z}$, denoted by $\vec{z}'$, is still an $N \times 1$ column vector. It can be shown (Wintz, 1972) that the mean square reconstruction error between the original vector $\vec{z}$ and the reconstructed vector $\vec{z}$ is given by

$$MSE_r = \sum_{i=L+1}^{N} \sigma_{y,i}^2 . \tag{4.16}$$

This equation indicates that the mean square reconstruction error equals the sum of variances of the discarded components. Note that although we discuss the reconstruction error here, we have not considered the quantization error and transmission error involved. Equation 4.15 implies that if, in the transformed vector $\vec{y}$, the first L components have their variances occupy a large percentage of the total variances, the mean square reconstruction error will not be large even though only the first L components are kept, i.e., the $(N - L)$ remaining components in the $\vec{y}$ are discarded. Quantizing and encoding only L components of vector $\vec{y}$ in the transform domain lead to higher coding efficiency. This is the basic idea behind transform coding.

4.1.3 Geometrical Interpretation

Transforming a set of statistically dependent data into another set of uncorrelated data, then discarding the insignificant transform coefficients (having small variances) illustrated above using the Hotelling transform, can be viewed as a statistical interpretation of transform coding. Here, we give a geometrical interpretation of transform coding. For this purpose, we use 2-D vectors instead of N-D vectors.

Consider a binary image of a car in Figure 4.1(a). Each pixel in the shaded object region corresponds to a 2-D vector with its two components being coordinates z_1 and z_2, respectively. Hence, the set of all pixels associated with the object forms a population of vectors. We can determine its mean vector and covariance matrix using Equations 4.12 and 4.13, respectively. We can then apply the Hotelling transform by using Equation 4.7. Figure 4.1(b) depicts the same object after the application of the Hotelling transform in the y_1-y_2 coordinate system. We notice that the origin of the new coordinate system is now located at the centroid of the binary object. Furthermore, the new coordinate system is aligned with the two eigenvectors of the covariance matrix $C_{\vec{z}}$.

As mentioned, the elements along the main diagonal of $C_{\vec{y}}$ (two eigenvalues of the $C_{\vec{y}}$ and $C_{\vec{z}}$) are the two variances of the two components of the $\vec{y}$ population. Since the covariance matrix

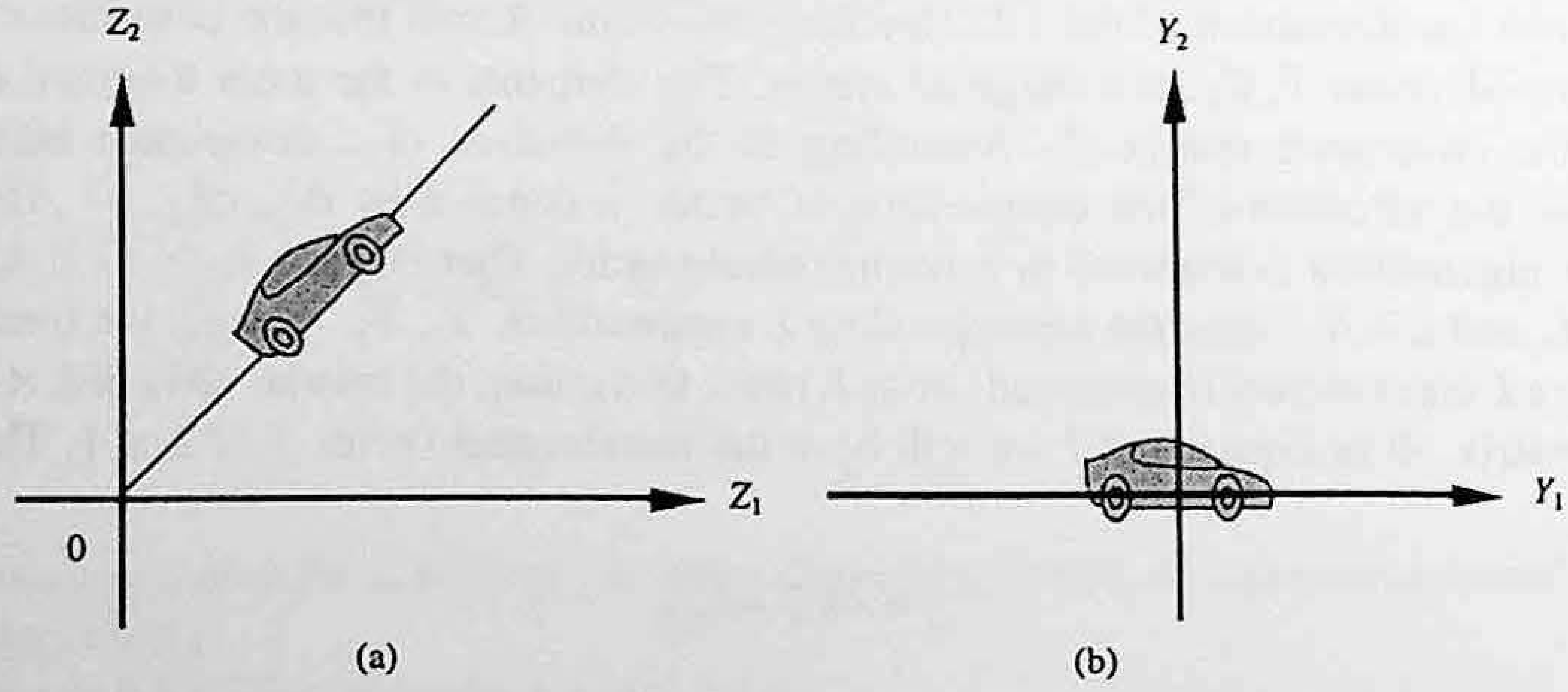


FIGURE 4.1 (a) A binary object in the z_1-z_2 coordinate system. (b) After the Hotelling transform, the object is aligned with its principal axes.

$C_{\vec{y}}$ is a diagonal matrix, the two components are uncorrelated after the transform. Since one variance (along the y_1 direction) is larger than the other (along the y_2 direction), it is possible for us to achieve higher coding efficiency by ignoring the component associated with the smaller variance without too much sacrifice of the reconstructed image quality.

It is noted that the alignment of the object with the eigenvectors of the covariance matrix is of importance in pattern recognition (Gonzalez and Woods, 1992).

4.1.4 Basis Vector Interpretation

Basis vector expansion is another interpretation of transform coding. For simplicity, in this subsection we assume a zero mean vector. Under this assumption, the Hotelling transform and its inverse transform become

$$\vec{y} = \Phi\vec{z} \tag{4.17}$$

$$\vec{z} = \Phi^T\vec{y} \tag{4.18}$$

Recall that the row vectors in the matrix Φ are the transposed eigenvectors of the covariance matrix $C_{\vec{z}}$. Therefore, Equation 4.18 can be written as

$$\vec{z} = \sum_{i=1}^{N} y_i\vec{e}_i . \tag{4.19}$$

In the above equation, we can view vector $\vec{z}$ as a linear combination of *basis vectors* $\vec{e}_i$, $i = 1,2,\cdots,N$. The components of the transformed vector $\vec{y}$, y_i, $i = 1,2,\cdots,N$ serve as coefficients in the linear combination, or weights in the weighted sum of basis vectors. The coefficient y_i, $i = 1,2,\cdots,N$ can be produced according to Equation 4.17:

$$y_i = \vec{e}_i^{\,T}\vec{z} . \tag{4.20}$$

That is, y_i is the *inner product* between vectors $\vec{e}_i$ and $\vec{z}$. Therefore, the coefficient y_i can be interpreted as the amount of correlation between the basis vector $\vec{e}_i$ and the original signal $\vec{z}$.

In the Hotelling transform the coefficients y_i, $i = 1,2,\cdots,N$ are uncorrelated. The variance of y_i can be arranged in a nonincreasing order. For $i > L$, the variance of the coefficient becomes insignificant. We can then discard these coefficients without introducing significant error in the linear combination of basis vectors and achieve higher coding efficiency.

In the above three interpretations of transform coding, we see that the linear unitary transform can provide the following two functions:

1. Decorrelate input data; i.e., transform coefficients are less correlated than the original data, and
2. Have some transform coefficients more significant than others (with large variance, eigenvalue, or weight in basis vector expansion) such that transform coefficients can be treated differently: some can be discarded, some can be coarsely quantized, and some can be finely quantized.

Note that the definition of *unitary* transform is given shortly in Section 4.2.1.3.

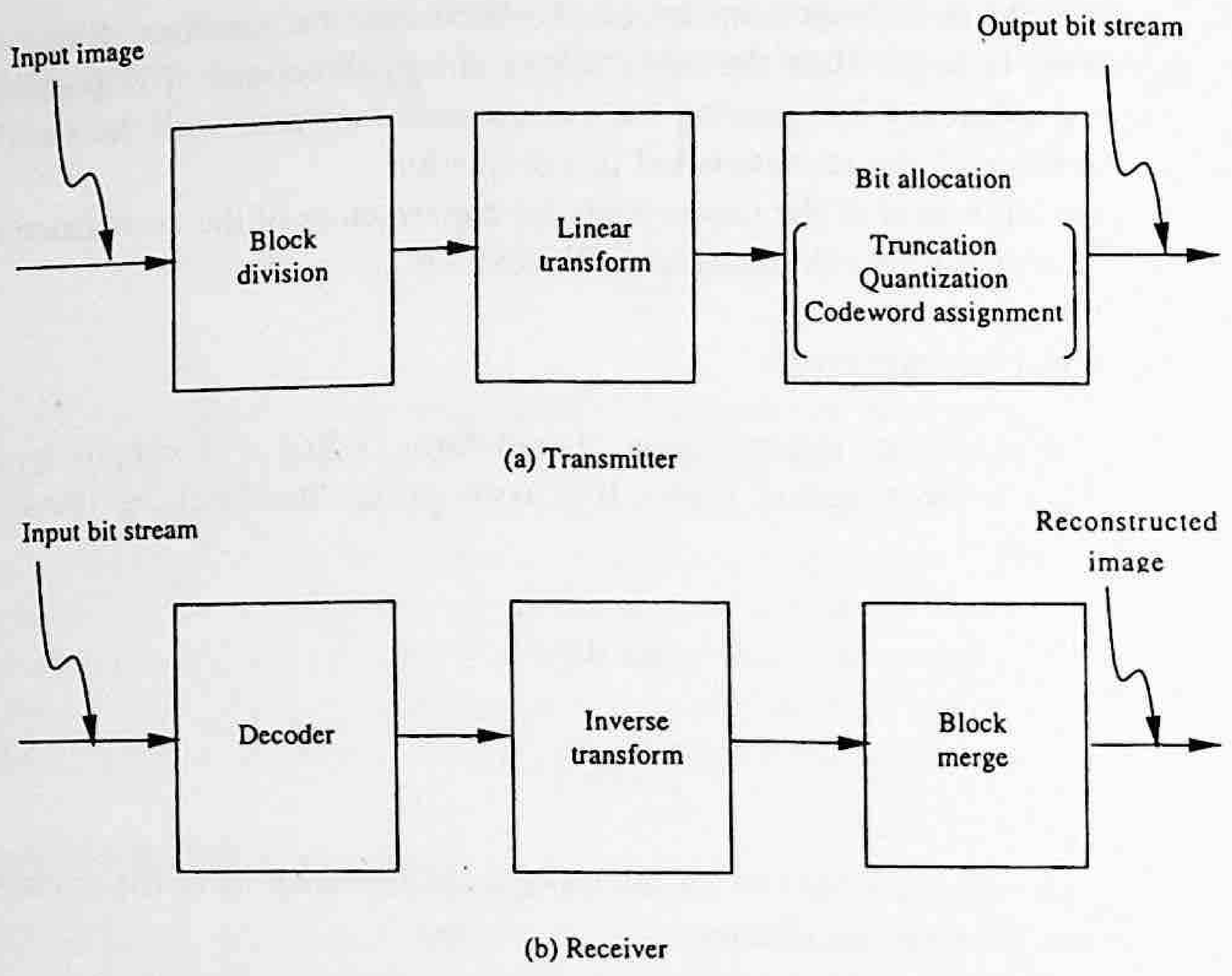


FIGURE 4.2 Block diagram of transform coding.

4.1.5 Procedures of Transform Coding

Prior to leaving this section, we summarize the procedures of transform coding. There are three steps in transform coding as shown in Figure 4.2. First, the input data (frame) are divided into blocks (subimages). Each block is then linearly transformed. The transformed version is then truncated, quantized, and encoded. These last three functions, which are discussed in Section 4.4, can be grouped and termed as bit allocation. The output of the encoder is a bitstream.

In the receiver, the bitstream is decoded and then inversely transformed to form reconstructed blocks. All the reconstructed blocks collectively produce a replica of the input image.

4.2 LINEAR TRANSFORMS

In this section, we first discuss a general formulation of a linear unitary 2-D image transform. Then, a basis image interpretation of TC is given.

4.2.1 2-D Image Transformation Kernel

There are two different ways to handle image transformation. In the first way, we convert a 2-D array representing a digital image into a 1-D array via row-by-row stacking, for example. That is, from the second row on, the beginning of each row in the 2-D array is cascaded to the end of its previous row. Then we transform this 1-D array using a 1-D transform. After the transformation, we can convert the 1-D array back to a 2-D array. In the second way, a 2-D transform is directly applied to the 2-D array corresponding to an input image, resulting in a transformed 2-D array. These two ways are essentially the same. It can be straightforwardly shown that the difference between the two is simply a matter of notation (Wintz, 1972). In this section, we use the second way to handle image transformation. That is, we work on 2-D image transformation.

Assume a digital image is represented by a 2-D array $g(x, y)$, where (x, y) is the coordinates of a pixel in the 2-D array, while g is the gray level value (also often called intensity or brightness) of the pixel. Denote the 2-D transform of $g(x, y)$ by $T(u, v)$, where (u, v) is the coordinates in the transformed domain. Assume that both $g(x, y)$ and $T(u, v)$ are a square 2-D array of $N \times N$; i.e., $0 \le x, y, u, v \le N - 1$.

The 2-D forward and inverse transforms are defined as

$$T(u,v) = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} g(x,y) f(x,y,u,v) \tag{4.21}$$

and

$$g(x,y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} T(u,v) i(x,y,u,v) \tag{4.22}$$

where $f(x, y, u, v)$ and $i(x, y, u, v)$ are referred to as the forward and inverse *transformation kernels*, respectively.

A few characteristics of transforms are discussed below.

4.2.1.1 Separability

A transformation kernel is called separable (hence, the transform is said to be separable) if the following conditions are satisfied.

$$f(x,y,u,v) = f_1(x,u) f_2(y,v), \tag{4.23}$$

and

$$i(x,y,u,v) = i_1(x,u) i_2(y,v). \tag{4.24}$$

Note that a 2-D separable transform can be decomposed into two 1-D transforms. That is, a 2-D transform can be implemented by a 1-D transform rowwise followed by another 1-D transform columnwise. That is,

$$T_1(x,v) = \sum_{y=0}^{N-1} g(x,y) f_2(y,v), \tag{4.25}$$

where $0 \le x, v \le N - 1$, and

$$T(u,v) = \sum_{x=0}^{N-1} T_1(x,v) f_1(x,u), \tag{4.26}$$

where $0 \le u, v \le N - 1$. Of course, the 2-D transform can also be implemented in a reverse order with two 1-D transforms, i.e., columnwise first, followed by rowwise. The counterparts of Equations 4.25 and 4.26 for the inverse transform can be derived similarly.

4.2.1.2 Symmetry

The transformation kernel is symmetric (hence, the transform is symmetric) if the kernel is separable and the following condition is satisfied:

$$f_1(y,v) = f_2(y,v). \tag{4.27}$$

That is, f_1 is functionally equivalent to f_2.

4.2.1.3 Matrix Form

If a transformation kernel is symmetric (hence, separable) then the 2-D image transform discussed above can be expressed compactly in the following matrix form. Denote an *image matrix* by G and $G = \{g_{i,j}\} = \{g(i-1, j-1)\}$. That is, a typical element (at the ith row and jth column) in the matrix G is the pixel gray level value in the 2-D array $g(x, y)$ at the same geometrical position. Note that the subtraction of one in the notation $g(i-1, j-1)$ comes from Equations 4.21 and 4.22. Namely, the indexes of a square 2-D image array are conventionally defined from 0 to N-1, while the indexes of a square matrix are from 1 to N. Denote the *forward transform matrix* by F and $F = \{f_{i,j}\} = \{f_1(i-1, j-1)\}$. We then have the following matrix form of a 2-D transform:

$$T = F^T GF \tag{4.28}$$

where T on the left-hand side of the equation denotes the matrix corresponding to the transformed 2-D array in the same fashion as that used in defining the G matrix. The inverse transform can be expressed as

$$G = I^T TI \tag{4.29}$$

where the matrix I is the *inverse transform matrix* and $I = \{i_{j,k}\} = \{i_1\,(j-1, k-1)\}$. The forward and inverse transform matrices have the following relation:

$$I = F^{-1} \tag{4.30}$$

Note that all of the matrices defined above, G, T, F, and I are of $N \times N$.

It is known that the discrete Fourier transform involves complex quantities. In this case, the counterparts of Equations 4.28, 4.29, and 4.30 become Equations 4.31, 4.32, and 4.33, respectively:

$$T = F^{*T} GF \tag{4.31}$$

$$G = I^{*T} TI \tag{4.32}$$

$$I = F^{-1} = F^{*T} \tag{4.33}$$

where * indicates complex conjugation. Note that the transform matrices F and I contain complex quantities and satisfy Equation 4.33. They are called unitary matrices and the transform is referred to as a unitary transform.

4.2.1.4 Orthogonality

A transform is said to be orthogonal if the transform matrix is orthogonal. That is,

$$F^T = F^{-1} \tag{4.34}$$

Note that an orthogonal matrix (orthogonal transform) is a special case of a unitary matrix (unitary transform), where only real quantities are involved. We will see that all the 2-D image transforms, presented in Section 4.3, are separable, symmetric, and unitary.

4.2.2 Basis Image Interpretation

Here we study the concept of *basis images* or *basis matrices.* Recall that we discussed basis vectors when we considered the 1-D transform. That is, the components of the transformed vector (also referred to as the transform coefficients) can be interpreted as the coefficients in the basis vector expansion of the input vector. Each coefficient is essentially the amount of correlation between the input vector and the corresponding basis vector. The concept of basis vectors can be extended to basis images in the context of 2-D image transforms.

Recall that the 2-D inverse transform introduced at the beginning of this section is defined as

$$g(x,y) = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} T(u,v)i(x,y,u,v) \tag{4.35}$$

where $0 \le x, y \le N-1$. This equation can be viewed as a *component* form of the inverse transform. As defined above in Section 4.2.1.3, the whole image $\{g(x, y)\}$ is denoted by the image matrix G of $N \times N$. We now denote the "image" formed by the inverse transformation kernel $\{i(x, y, u, v), 0 \le x, y \le N-1\}$ as a 2-D array $I_{u,v}$ of $N \times N$ for a specific pair of (u, v) with $0 \le u, v \le N-1$. Recall that a digital image can be represented by a 2-D array of gray level values. In turn the 2-D array can be arranged into a matrix. Namely, we treat the following three: a digital image, a 2-D array (with proper resolution), and a matrix (with proper indexing), interchangeably. We then have

$$I_{u,v} = \begin{bmatrix} i(0,0,u,v) & i(0,1,u,v) & \cdots & \cdots & i(0,N-1,u,v) \\ i(1,0,u,v) & i(1,1,u,v) & \cdots & \cdots & i(1,N-1,u,v) \\ \vdots & \vdots & \cdots & \cdots & \vdots \\ \vdots & \vdots & \cdots & \cdots & \vdots \\ i(N-1,0,u,v) & i(N-1,1,u,v) & \cdots & \cdots & i(N-1,N-1,u,v) \end{bmatrix} \tag{4.36}$$

The 2-D array $I_{u,v}$ is referred to as a basis image. There are N^2 basis images in total since $0 \le u,v \le N-1$. The inverse transform expressed in Equation 4.35 can then be written in a *collective* form as

$$G = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} T(u,v)I_{u,v}. \tag{4.37}$$

We can interpret this equation as a series expansion of the original image G into a set of N^2 basis images $I_{u,v}$. The transform coefficients $T(u,v)$, $0 \le u, v \le N-1$, become the coefficients of the expansion. Alternatively, the image G is said to be a weighted sum of basis images. Note that, similar to the 1-D case, the coefficient or the weight $T(u,v)$ is a correlation measure between the image G and the basis image $I_{u,v}$ (Wintz, 1972).

Note that basis images have nothing to do with the input image. Instead, it is completely defined by the transform itself. That is, basis images are the attribute of 2-D image transforms. Different transforms have different sets of basis images.

The motivation behind transform coding is that with a proper transform, hence, a proper set of basis images, the transform coefficients are more independent than the gray scales of the original input image. In the ideal case, the transform coefficients are statistically independent. We can then optimally encode the coefficients independently, which can make coding more efficient and simple. As pointed out in (Wintz, 1972), however, this is generally impossible because of the following two reasons. First, it requires the joint probability density function of the N^2 pixels, which have not been deduced from basic physical laws and cannot be measured. Second, even if the joint probability density functions were known, the problem of devising a reversible transform that can generate independent coefficients is unsolved. The optimum linear transform we can have results in uncorrelated coefficients. When Gaussian distribution is involved, we can have independent transform coefficients. In addition to the uncorrelatedness of coefficients, the variance of the coefficients varies widely. Insignificant coefficients can be ignored without introducing significant distortion in the reconstructed image. Significant coefficients can be allocated more bits in encoding. The coding efficiency is thus enhanced.

As shown in Figure 4.3, TC can be viewed as expanding the input image into a set of basis images, then quantizing and encoding the coefficients associated with the basis images separately. At the receiver the coefficients are reconstructed to produce a replica of the input image. This strategy is similar to that of subband coding, which is discussed in Chapter 8. From this point of view, transform coding can be considered a special case of subband coding, though transform coding was devised much earlier than subband coding.

It is worth mentioning an alternative way to define basis images. That is, a basis image with indexes (u, v), $I_{u,v}$, of a transform can be constructed as the *outer product* of the uth basis vector, $\vec{b}_u$, and the vth basis vector, $\vec{b}_v$, of the transform. The basis vector, $\vec{b}_u$, is the uth column vector of the inverse transform matrix I (Jayant and Noll, 1984). That is,

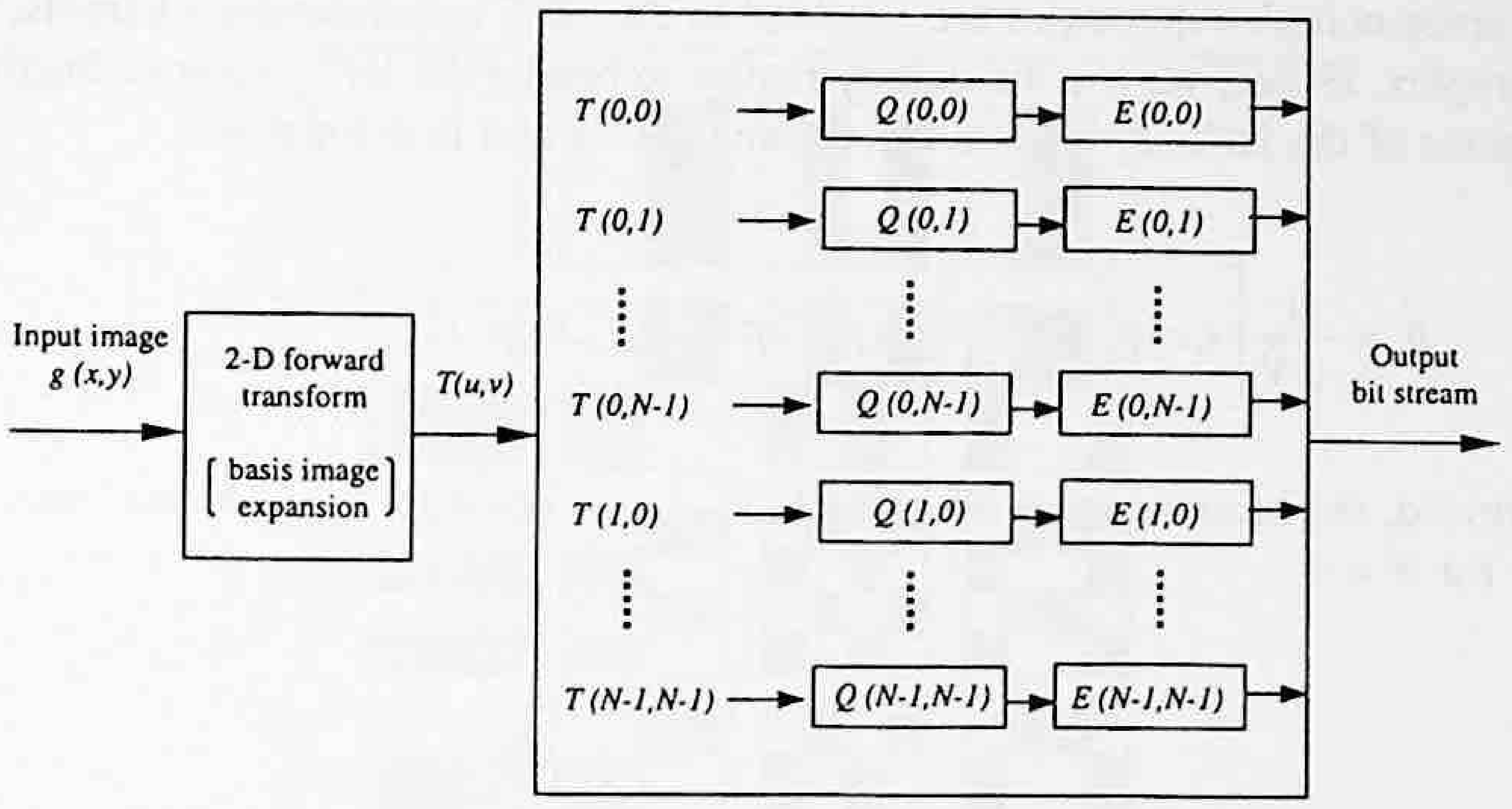
$$I_{u,v} = \vec{b}_u \vec{b}_v^{\,T}. \tag{4.38}$$

4.2.3 Subimage Size Selection

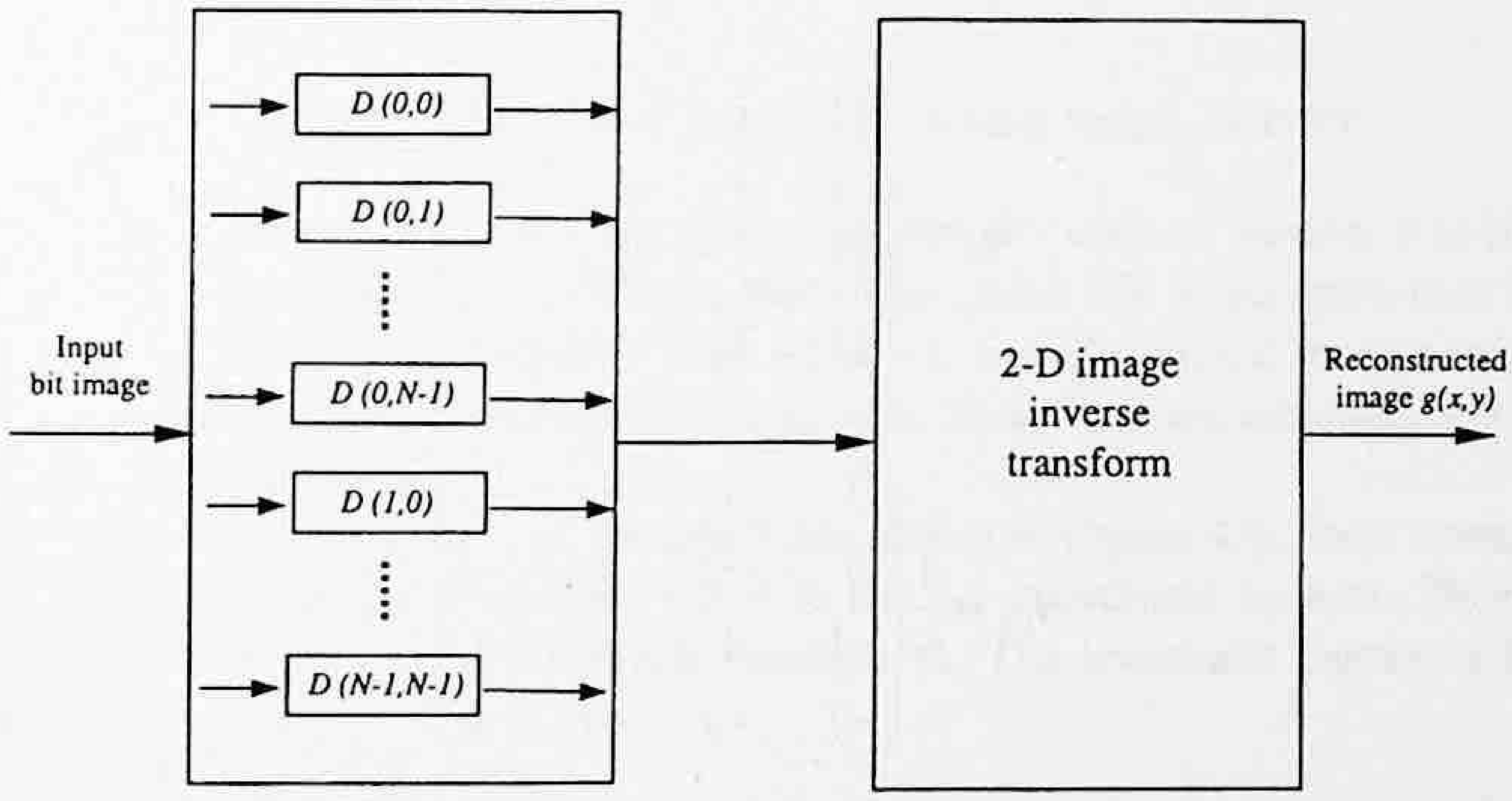
The selection of subimage (block) size, N, is important. Normally, the larger the size the more decorrelation the transform coding can achieve. It has been shown, however, that the correlation between image pixels becomes insignificant when the distance between pixels becomes large, e.g., it exceeds 20 pixels (Habibi, 1971a). On the other hand, a large size causes some problems. In adaptive transform coding, a large block cannot adapt to local statistics well. As will be seen later in this chapter, a transmission error in transform coding affects the whole associated subimage. Hence a large size implies a possibly severe effect of transmission error on reconstructed images. As will be shown in video coding (Section III and Section IV), transform coding is used together with motion-compensated coding. Consider that large block size is not used in motion estimation; subimage sizes of 4, 8, and 16 are used most often. In particular, $N = 8$ is adopted by the international still image coding standard, JPEG, as well as video coding standards H.261, H.263, MPEG 1, and MPEG 2.

4.3 TRANSFORMS OF PARTICULAR INTEREST

Several commonly used image transforms are discussed in this section. They include the discrete Fourier transform, the discrete Walsh transform, the discrete Hadamard transform, and the discrete Cosine and Sine transforms. All of these transforms are symmetric (hence, separable as well),



(a) Transmitter



(b) Receiver

FIGURE 4.3 Basis image interpretation of TC (Q: quantizer, E: encoder, D: decoder).

unitary, and reversible. For each transform, we define its transformation kernel and discuss its basis images.

4.3.1 Discrete Fourier Transform (DFT)

The DFT is of great importance in the field of digital signal processing. Owing to the fast Fourier transform (FFT) based on the algorithm developed in (Cooley, 1965), the DFT is widely utilized for various tasks of digital signal processing. It has been discussed in many signal and image processing texts. Here we only define it by using the transformation kernel just introduced above. The forward and inverse transformation kernels of the DFT are

$$f(x,y,u,v)=\frac{1}{N}\exp\{-j2\pi(xu+yv)/N\} \tag{4.39}$$

and

$$i(x,y,u,v)=\frac{1}{N}\exp\{j2\pi(xu+yv)/N\} \tag{4.40}$$

Clearly, since complex quantities are involved in the DFT transformation kernels, the DFT is generally complex. Hence, we use the unitary matrix to handle the DFT (refer to Section 4.2.1.3). The basis vector of the DFT $\vec{b}_u$ is an $N \times 1$ column vector and is defined as

$$\vec{b}_u = \frac{1}{\sqrt{N}}\left[1, \exp\left(j2\pi\frac{u}{N}\right), \exp\left(j2\pi\frac{2u}{N}\right), \cdots, \exp\left(j2\pi\left(\frac{(N-1)u}{N}\right)\right)\right]^T \tag{4.41}$$

As mentioned, the basis image with index (u,v), $I_{u,v}$, is equal to $\vec{b}_u \vec{b}_u^T$. A few basis images are listed below for $N = 4$.

$$I_{0,0} = \frac{1}{4}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \tag{4.42}$$

$$I_{0,1} = \frac{1}{4}\begin{pmatrix} 1 & j & -1 & -j \\ 1 & j & -1 & -j \\ 1 & j & -1 & -j \\ 1 & j & -1 & -j \end{pmatrix} \tag{4.43}$$

$$I_{1,2} = \frac{1}{4}\begin{pmatrix} 1 & 1 & 1 & -j \\ j & -j & j & -j \\ -1 & 1 & -1 & 1 \\ -j & -j & -j & j \end{pmatrix} \tag{4.44}$$

$$I_{3,3} = \frac{1}{4}\begin{pmatrix} 1 & -j & -1 & j \\ -j & -1 & j & 1 \\ -1 & j & 1 & -j \\ j & 1 & -j & -1 \end{pmatrix} \tag{4.45}$$

4.3.2 Discrete Walsh Transform (DWT)

The transformation kernels of the DWT (Walsh, 1923) are defined as

$$f(x,y,u,v) = \frac{1}{N}\prod_{i=0}^{n-1}\left[(-1)^{p_i(x)p_{n-1-i}(u)}(-1)^{p_i(y)p_{n-1-i}(v)}\right] \tag{4.46}$$

and

$$i(x,y,u,v) = f(x,y,u,v). \tag{4.47}$$

where $n = \log_2 N$, $p_i(\text{arg})$ represents the ith bit in the natural binary representation of the arg, the oth bit corresponds to the least significant bit, and the $(n\text{-}1)$th bit corresponds to the most significant

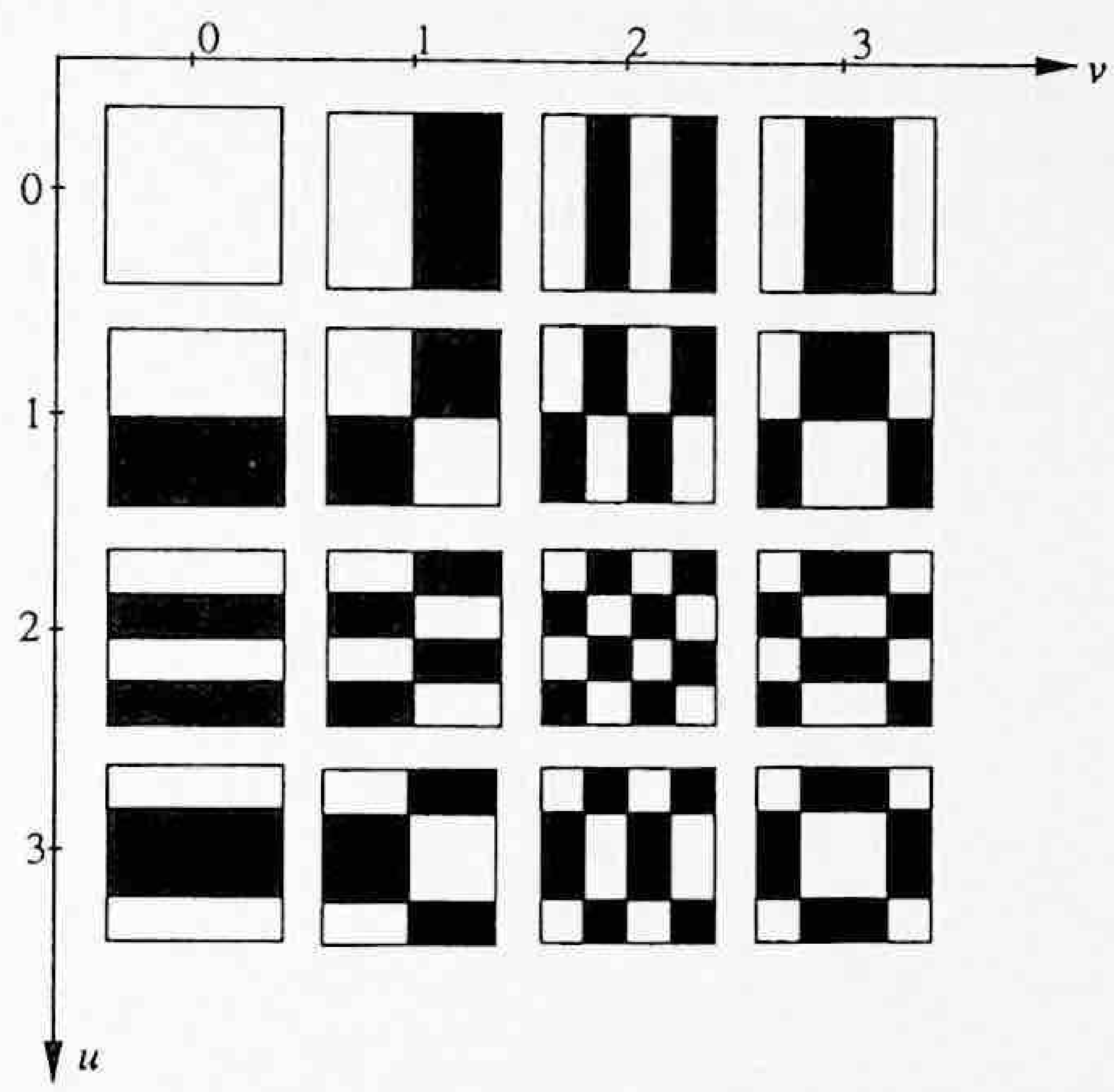


FIGURE 4.4 When N = 4: a set of the 16 basis images of DWT.

bit. For instance, consider $N = 16$, then $n = 4$. The natural binary code of number 8 is 1000. Hence, $p_0(8) = p_1(8) = p_2(8) = 0$, and $p_3(8) = 1$. We see that if the factor $1/N$ is put aside then the forward transformation kernel is always an integer: either +1 or −1. In addition, the inverse transformation kernel is the same as the forward transformation kernel. Therefore, we conclude that the implementation of the DWT is simple.

When $N = 4$, the 16 basis images of the DWT are shown in Figure 4.4. Each corresponds to a specific pair of (u, v) and is of resolution 4×4 in the x-y coordinate system. They are binary images, where the bright represents +1, while the dark −1. The transform matrix of the DWT is shown below for $N = 4$.

$$F = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \tag{4.48}$$

4.3.3 Discrete Hadamard Transform (DHT)

The DHT (Hadamard, 1893) is closely related to the DWT. This can be seen from the following definition of the transformation kernels.

$$f(x, y, u, v) = \frac{1}{N}\prod_{i=0}^{n}\left[(-1)^{p_i(x)p_i(u)}(-1)^{p_i(y)p_i(v)}\right] \tag{4.49}$$

and

$$i(x, y, u, v) = f(x, y, u, v) \tag{4.50}$$

where the definitions of n, i, and $p_i(\text{arg})$ are the same as in the DWT. For this reason, the term Walsh-Hadamard transform (DWHT) is frequently used to represent either of the two transforms.

When N is a power of 2, the transform matrices of the DWT and DHT have the same row (or column) vectors except that the order of row (or column) vectors in the matrices are different. This is the only difference between the DWT and DHT under the circumstance $N = 2^n$. Because of this difference, while the DWT can be implemented by using the FFT algorithm with a straightforward modification, the DHT needs more work to use the FFT algorithm. On the other hand, the DHT possesses the following recursive feature, while the DWT does not:

$$F_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{4.51}$$

and

$$F_{2N} = \begin{bmatrix} F_N & F_N \\ F_N & -F_N \end{bmatrix} \tag{4.52}$$

where the subscripts indicate the size of the transform matrices. It is obvious that the transform matrix of the DHT can be easily derived by using the recursion.

Note that the number of sign changes between consecutive entries in a row (or a column) of a transform matrix (from positive to negative and from negative to positive) is known as *sequency*. It is observed that the sequency does not monotonically increase as the order number of rows (or columns) increases in the DHT. Since sequency bears some similarity to frequency in the Fourier transform, sequency is desired as an increasing function of the order number of rows (or columns). This is realized by the *ordered* Hadamard transform (Gonzalez, 1992).

The transformation kernel of the ordered Hadamard transform is defined as

$$f(x, y, u, v) = \frac{1}{N} \prod_{i=0}^{N-1} \left[(-1)^{p_i(x) d_i(u)} (-1)^{p_i(y) d_i(v)} \right] \tag{4.53}$$

where the definitions of i, $p_i(\text{arg})$ are the same as defined above for the DWT and DHT. The $d_i(\text{arg})$ is defined as below.

$$\begin{aligned} d_0(\text{arg}) &= b_{n-1}(\text{arg}) \\ d_1(\text{arg}) &= b_{n-1}(\text{arg}) + b_{n-2}(\text{arg}) \\ d_{n-1}(\text{arg}) &= b_1(\text{arg}) + b_0(\text{arg}) \end{aligned} \tag{4.54}$$

The 16 basis images of the ordered Hadamard transform are shown in Figure 4.5 for $N = 4$. It is observed that the variation of the binary basis images becomes more frequent monotonically when u and v increase. Also we see that the basis image expansion is similar to the frequency expansion of the Fourier transform in the sense that an image is decomposed into components with different variations. In transform coding, these components with different coefficients are treated differently.

4.3.4 Discrete Cosine Transform (DCT)

The DCT is the most commonly used transform for image and video coding.

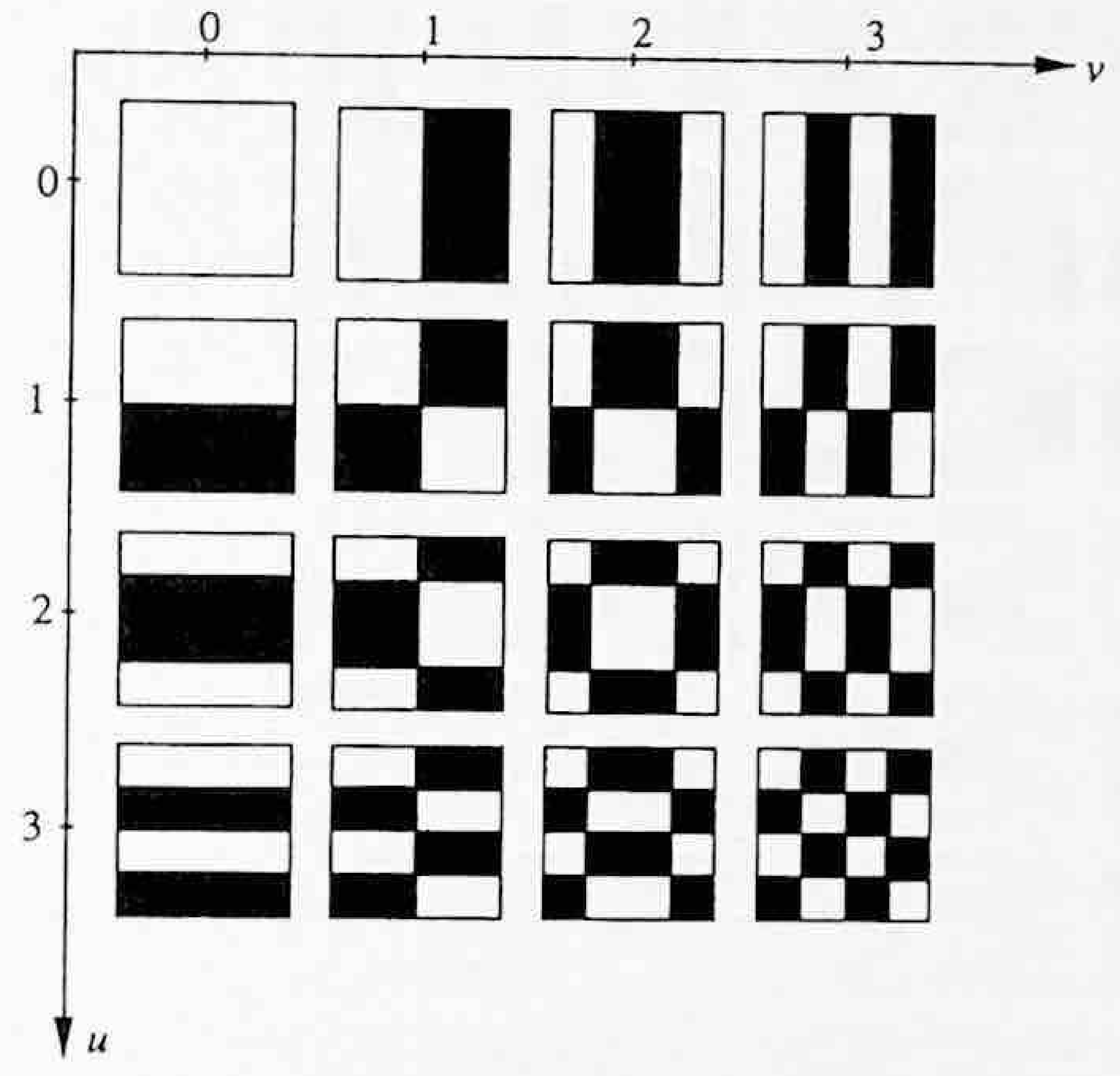


FIGURE 4.5 When N = 4: a set of the 16 basis images of the ordered DHT.

4.3.4.1 Background

The DCT, which plays an extremely important role in image and video coding, was established by Ahmed et al. (1974). There, it was shown that the *basis member* $\cos[(2x + 1)u\pi/2N]$ is the uth Chebyshev polynomial $T_U(\xi)$ evaluated at the xth zero of $T_N(\xi)$. Recall that the Chebyshev polynomials are defined as

$$T_0(\xi) = 1/\sqrt{2} \tag{4.55}$$

$$T_K(\xi) = \cos\left[k\cos^{-1}(\xi)\right] \tag{4.56}$$

where $T_K(\xi)$ is the kth order Chebyshev polynomial and it has k zeros, starting from the *1*st zero to the kth zero. Furthermore, it was demonstrated that the basis vectors of 1-D DCT provide a good approximation to the eigenvectors of the class of Toeplitz matrices defined as

$$\begin{bmatrix} 1 & \rho & \rho^2 & \cdots & \rho^{N-1} \\ \rho & 1 & \rho & \cdots & \rho^{N-2} \\ \rho^2 & \rho & 1 & \cdots & \rho^{N-3} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ \rho^{N-1} & \rho^{N-2} & \rho^{N-3} & \cdots & 1 \end{bmatrix}, \tag{4.57}$$

where $0 < \rho < 1$.

4.3.4.2 Transformation Kernel

The transformation kernel of the 2-D DCT can be extended straightforwardly from that of 1-D DCT as follows:

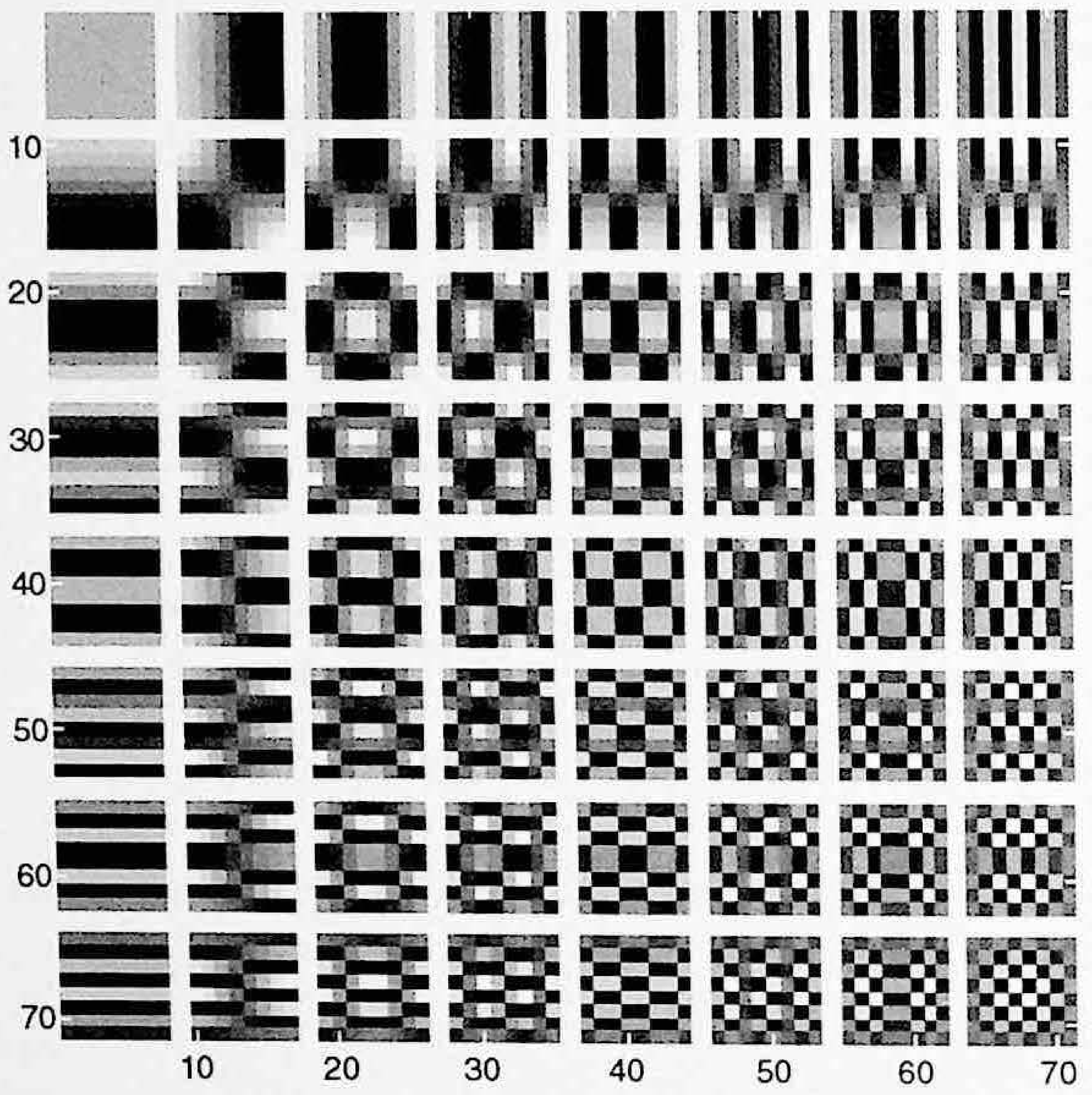


FIGURE 4.6 When N = 8: a set of the 64 basis images of the DCT.

$$f(x,y,u,v) = C(u)C(v)\cos\left(\frac{(2x+1)u\pi}{2N}\right)\cos\left(\frac{(2y+1)v\pi}{2N}\right) \tag{4.58}$$

where

$$C(u) = \begin{cases} \sqrt{\frac{1}{N}} & for \quad u = 0 \\ \sqrt{\frac{2}{N}} & for \quad u = 1,2,\cdots,N-1 \end{cases} \tag{4.59}$$

$$i(x,y,u,v) = f(x,y,u,v). \tag{4.60}$$

Note that the $C(v)$ is defined the same way as in Equation 4.59. The 64 basis images of the DCT are shown in Figure 4.6 for $N = 8$.

4.3.4.3 Relationship with DFT

The DCT is closely related to the DFT. This can be examined from an alternative method of defining the DCT. It is known that applying the DFT to an N-point sequence $g_N(n)$, $n = 0,1,\cdots,N-1$, is equivalent to the following:

1. Repeating $g_N(n)$ every N points, form a periodic sequence, $\tilde{g}_N(n)$, with a fundamental period N. That is,

$$\tilde{g}_N(n) = \sum_{i=-\infty}^{\infty} g_N(n - iN). \tag{4.61}$$

2. Determine the Fourier series expansion of the periodic sequence $\tilde{g}_N(n)$. That is, determine all the coefficients in the Fourier series which are known to be periodic with the same fundamental period N.
3. Truncate the sequence of the Fourier series coefficients so as to have the same support as that of the given sequence $g_N(n)$. That is, only keep the N coefficients with indexes $0, 1, \cdots, N-1$ and set all the others to equal zero. These N Fourier series coefficients form the DFT of the given N-point sequence $g_N(n)$.

An N-point sequence $g_N(n)$ and the periodic sequence $\tilde{g}_N(n)$, generated from $g_N(n)$, are shown in Figure 4.7(a) and (b), respectively. In summary, the DFT can be viewed as a correspondence between two periodic sequences. One is the periodic sequence $\tilde{g}_N(n)$, which is formed by periodically repeating $g_N(n)$. The other is the periodic sequence of Fourier series coefficients of $\tilde{g}_N(n)$.

The DCT of an N-point sequence is obtained via the following three steps:

1. Flip over the given sequence with respect to the end point of the sequence to form a $2N$-point sequence, $g_{2N}(n)$, as shown in Figure 4.7(c). Then form a periodic sequence $\tilde{g}_{2N}(n)$, shown in Figure 4.7(d), according to

$$\tilde{g}_{2N}(n) = \sum_{i=-\infty}^{\infty} g_{2N}(n - 2iN) \tag{4.62}$$

2. Find the Fourier series coefficients of the periodic sequences $\tilde{g}_{2N}(n)$.
3. Truncate the resultant periodic sequence of the Fourier series coefficients to have the support of the given finite sequence $g_N(n)$. That is, only keep the N coefficients with indexes $0, 1, \cdots, N-1$ and set all the others to equal zero. These N Fourier series coefficients form the DCT of the given N-point sequence $g_N(n)$.

A comparison between Figure 4.7(b) and (d) reveals that the periodic sequence $\tilde{g}_N(n)$ is not smooth. There usually exist discontinuities at the beginning and end of each period. These end-head discontinuities cause a high-frequency distribution in the corresponding DFT. On the contrary, the periodic sequence $\tilde{g}_{2N}(n)$ does not have this type of discontinuity due to flipping over the given finite sequence. As a result, there is no high-frequency component corresponding to the end-head discontinuities. Hence, the DCT possesses better energy compaction in the low frequencies than the DFT. By better energy compaction, we mean more energy is compacted in a fraction of transform coefficients. For instance, it is known that the most energy of an image is contained in a small region of low frequency in the DFT domain. Vivid examples can be found in (Gonzalez and Woods, 1992). In terms of energy compaction, when compared with the Karhunen-Loeve transform (the Hotelling transform is its discrete version), which is known as the optimal, the DCT is the best among the DFT, DWT, DHT, and discrete Haar transform.

Besides this advantage, the DCT can be implemented using the FFT. This can be seen from the above discussion. There, it has been shown that the DCT of an N-point sequence, $g_N(n)$, can be obtained from the DFT of the $2N$-point sequence $g_{2N}(n)$. Furthermore, the even symmetry

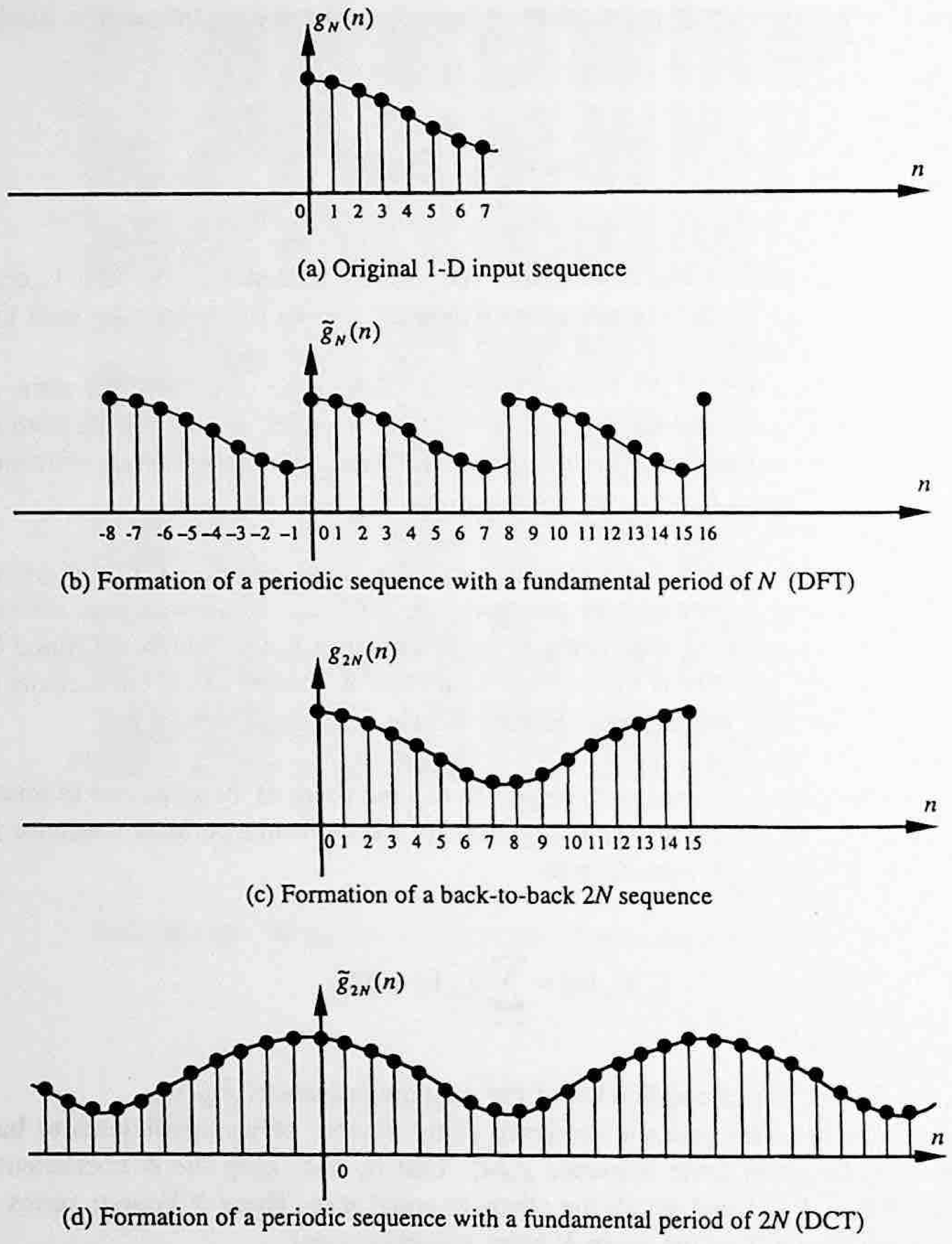

$g_N(n)$
n
0 1 2 3 4 5 6 7
(a) Original 1-D input sequence
$\tilde{g}_N(n)$
n
-8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
(b) Formation of a periodic sequence with a fundamental period of N (DFT)
$g_{2N}(n)$
n
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
(c) Formation of a back-to-back $2N$ sequence
$\tilde{g}_{2N}(n)$
n
0
(d) Formation of a periodic sequence with a fundamental period of $2N$ (DCT)

FIGURE 4.7 An example to illustrate the differences and similarities between DFT and DCT.

in $\tilde{g}_{2N}(n)$ makes the computation required for the DCT of an N-point equal to that required for the DFT of the N-point sequence. Because of these two merits, the DCT is the most popular image transform used in image and video coding nowadays.

4.3.5 Performance Comparison

In this subsection, we compare the performance of a few commonly used transforms in terms of energy compaction, mean square reconstruction error, and computational complexity.

4.3.5.1 Energy Compaction

Since all the transforms we discussed are symmetric (hence separable) and unitary, the matrix form of the 2-D image transform can be expressed as $T = F^T GF$ as discussed in Section 4.2.1.3. In the 1-D case, the transform matrix F is the counterpart of the matrix Φ discussed in the Hotelling transform. Using the F, one can transform a 1-D column vector $\vec{z}$ into another 1-D column vector $\vec{y}$. The components of the vector $\vec{y}$ are transform coefficients. The variances of these transform coefficients, and therefore the signal energy associated with the transform coefficients, can be arranged in a nondecreasing order. It can be shown that the total energy before and after the transform remains the same. Therefore, the more energy compacted in a fraction of total coefficients, the better energy compaction the transform has. One measure of energy compaction is the *transform coding gain* G_{TC}, which is defined as the ratio between the arithmetic mean and the geometric mean of the variances of all the components in the transformed vector (Jayant, 1984).

$$G_{TC} = \frac{\frac{1}{N}\sum_{i=0}^{N-1}\sigma_i^2}{\left(\prod_{i=0}^{N-1}\sigma_i^2\right)^{\frac{1}{N}}} \tag{4.63}$$

A larger G_{TC} indicates higher energy compaction. The transform coding gains for a first-order autoregressive source with $\rho = 0.95$ achieved by using the DCT, DFT, and KLT was reported in (Zelinski and Noll, 1975; Jayant and Noll, 1984). The transform coding gain afforded by the DCT compares very closely to that of the optimum KLT.

4.3.5.2 Mean Square Reconstruction Error

The performance of the transforms can be compared in terms of the mean square reconstruction error as well. This was mentioned in Section 4.1.2 when we provided a statistical interpretation for transform coding. That is, after arranging all the N transformed coefficients according to their variances in a nonincreasing order, if $L < N$ and we discard the last $(N - L)$ coefficients to reconstruct the original input signal $\vec{z}$ (similar to what we did with the Hotelling transform), then the mean square reconstruction error is

$$MSE_r = E\left[\|\vec{z} - \vec{z}'\|^2\right] = \sum_{i=L+1}^{N}\sigma_i^2, \tag{4.64}$$

where $\vec{z}'$ denotes the reconstructed vector. Note that in the above-defined mean square reconstruction error, the quantization error and transmission error have not been included. Hence, it is sometimes referred to as the mean square approximation error. Therefore it is desired to choose a transform so that the transformed coefficients are "more independent" and more energy is concentrated in the first L coefficients. Then it is possible to discard the remaining coefficients to save coding bits without causing significant distortion in input signal reconstruction.

In terms of the mean square reconstruction error, the performance of the DCT, KLT, DFT, DWT, and discrete Haar transform for the 1-D case was reported in Ahmed et al. (1974). The variances of the 16 transform coefficients are shown in Figure 4.8 when $N = 16$, $\rho = 0.95$. Note that N stands for the dimension of the 1-D vector, while the parameter ρ is shown in the Toeplitz matrix (refer to Equation 4.57). We can see that the DCT compares most closely to the KLT, which is known to be optimum.

Note that the unequal variance distribution among transform coefficients has also found application in the field of pattern recognition. Similar results to those in Ahmed et al. (1974) for the DFT, DWT, and Haar transform were reported in (Andrews, 1971).

A similar analysis can be carried out for the 2-D case (Wintz, 1972). Recall that an image $g(x, y)$ can be expressed as a weighted sum of basis images $I_{u,v}$. That is,

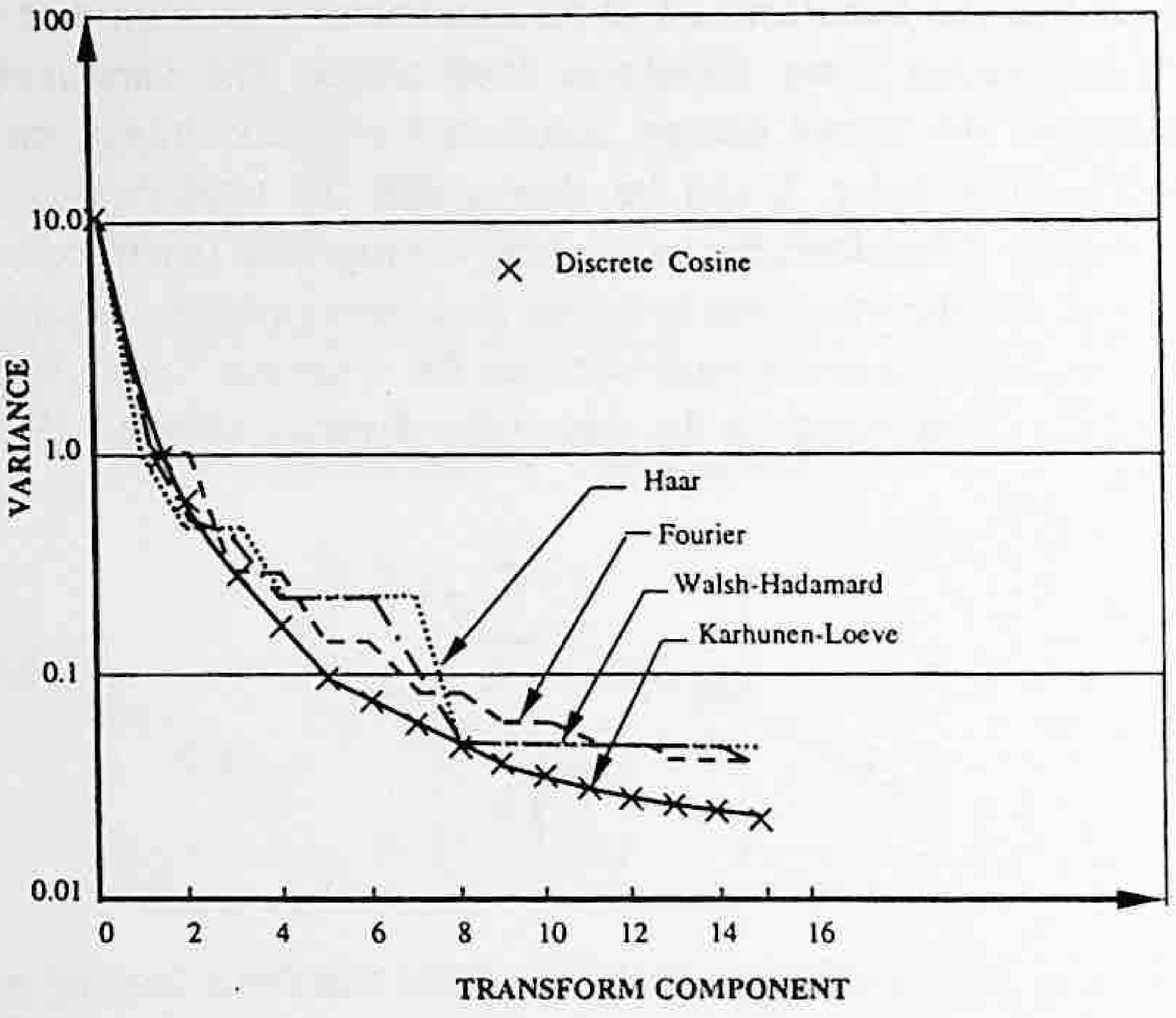


FIGURE 4.8 Transform coefficient variances when $N = 16$, $\rho = 0.95$. (From Ahmed, N. et al., *IEEE Trans. Comput.*, 90, 1974. With permission.)

$$G = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} T(u,v) I_{u,v} \tag{4.65}$$

where the weights are transform coefficients. We arrange the coefficients according to their variances in a nonincreasing order. For some choices of the transform (hence basis images), the coefficients become insignificant after the first L terms, and the image can be approximated well by truncating the coefficients after L. That is,

$$G = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} T(u,v) I_{u,v} \approx \sum_{u=0}^{L} \sum_{v=0} T(u,v) I_{u,v} \tag{4.66}$$

The mean square reconstruction error is given by

$$MSE_r = \sum_{L}^{N-1} \sum^{N-1} \sigma^2_{u,v} \tag{4.67}$$

A comparison among the KLT, DHT, and DFT in terms of the mean square reconstruction error for 2-D array of 16 × 16 (i.e., 256 transform coefficients) was reported in (Figure 5, Wintz, 1972). Note that the discrete KLT is image dependent. In the comparison, the KLT is calculated with respect to an image named "Cameraman." It shows that while the KLT achieves the best performance, the other transforms perform closely.

In essence, the criteria of mean square reconstruction error and energy compaction are closely related. It has been shown that the discrete Karhunen transform (KLT), also known as the Hotelling transform, is the optimum in terms of energy compaction and mean square reconstruction error. The DWT, DHT, DFT, and DCT are close to the optimum (Wintz, 1972; Ahmed et al., 1974); however, the DCT is the best among these several *suboptimum* transforms.

Note that the performance comparison among various transforms in terms of bit rate vs. distortion in the reconstructed image was reported in (Pearl et al., 1972; Ahmed et al., 1974). The same conclusion was drawn. That is, the KLT is optimum, while the DFT, DWT, DCT, and Haar transforms are close in performance. Among the suboptimum transforms, the DCT is the best.

4.3.5.3 Computational Complexity

Note that while the DWT, DHT, DFT, and DCT are input image independent, the discrete KLT (the Hotelling transform) is input dependent. More specifically, the row vectors of the Hotelling transform matrix are transposed eigenvectors of the covariance matrix of the input random vector. So far there is no fast transform algorithm available. This computational complexity prohibits the Hotelling transform from practical usage. It can be shown that the DWT, DFT, and DCT can be implemented using the FFT algorithm.

4.3.5.4 Summary

As pointed out above, the DCT is the best among the suboptimum transforms in terms of energy compaction. Moreover, the DCT can be implemented using the FFT. Even though a $2N$-point sequence is involved, the even symmetry makes the computation involved in the N-point DCT equivalent to that of the N-point FFT. For these two reasons, the DCT finds the widest application in image and video coding.

4.4 BIT ALLOCATION

As shown in Figure 4.2, in transform coding, an input image is first divided into blocks (subimages). Then a 2-D linear transform is applied to each block. The transformed blocks go through truncation, quantization, and codeword assignment. The last three functions: truncation, quantization, and codeword assignment, are combined and called bit allocation.

From the previous section, it is known that the applied transform decorrelates subimages. Moreover, it redistributes image energy in the transform domain in such a way that most of the energy is compacted into a small fraction of coefficients. Therefore, it is possible to discard the majority of transform coefficients without introducing significant distortion.

As a result, we see that in transform coding there are mainly three types of errors involved. One is due to truncation. That is, the majority of coefficients are truncated to zero. Others come from quantization. (Note that truncation can also be considered a special type of quantization). Transmission errors are the third type of error. Recall that the mean square reconstruction error discussed in Section 4.3.5.2 is in fact only related to truncation error. For this reason, it was referred to more precisely as a mean square approximation error. In general, the reconstruction error, i.e., the error between the original image signal and the reconstructed image at the receiver, includes three types of errors: truncation error, quantization error, and transmission error.

There are two different ways to truncate transform coefficients. One is called *zonal coding*, while the other is *threshold coding*. They are discussed below.

4.4.1 Zonal Coding

In zonal coding, also known as *zonal sampling*, a zone in the transformed block is predefined according to a statistical average obtained from many blocks. All transform coefficients in the zone are retained, while all coefficients outside the zone are set to zero. As mentioned in Section 4.3.5.1, the total energy of the image remains the same after applying the transforms discussed there. Since it is known that the DC and low-frequency AC coefficients of the DCT occupy most of the energy, the zone is located in the top-left portion of the transformed block when the transform coordinate

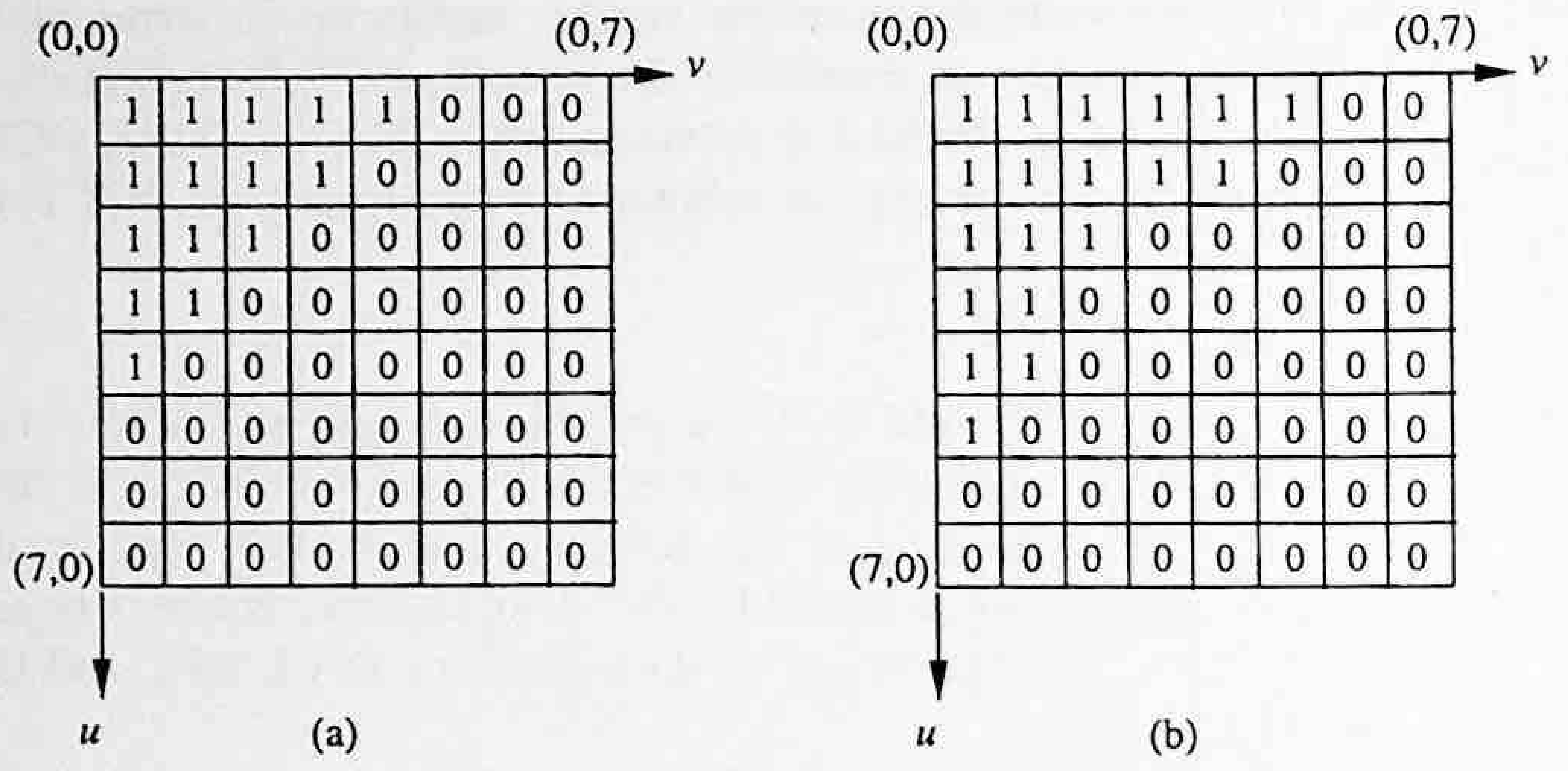


FIGURE 4.9 Two illustrations of zonal coding.

system is set conventionally. (Note that by DC we mean $u = v = 0$. By AC we mean u and v do not equal zero simultaneously.) That is, the origin is at the top-left corner of the transformed block. Two typical zones are shown in Figure 4.9. The simplest uniform quantization with natural binary coding can be used to quantize and encode the retained transform coefficients. With this simple technique, there is no overhead side information that needs to be sent to the receiver, since the structure of the zone, the scheme of the quantization, and encoding are known at both the transmitter and receiver.

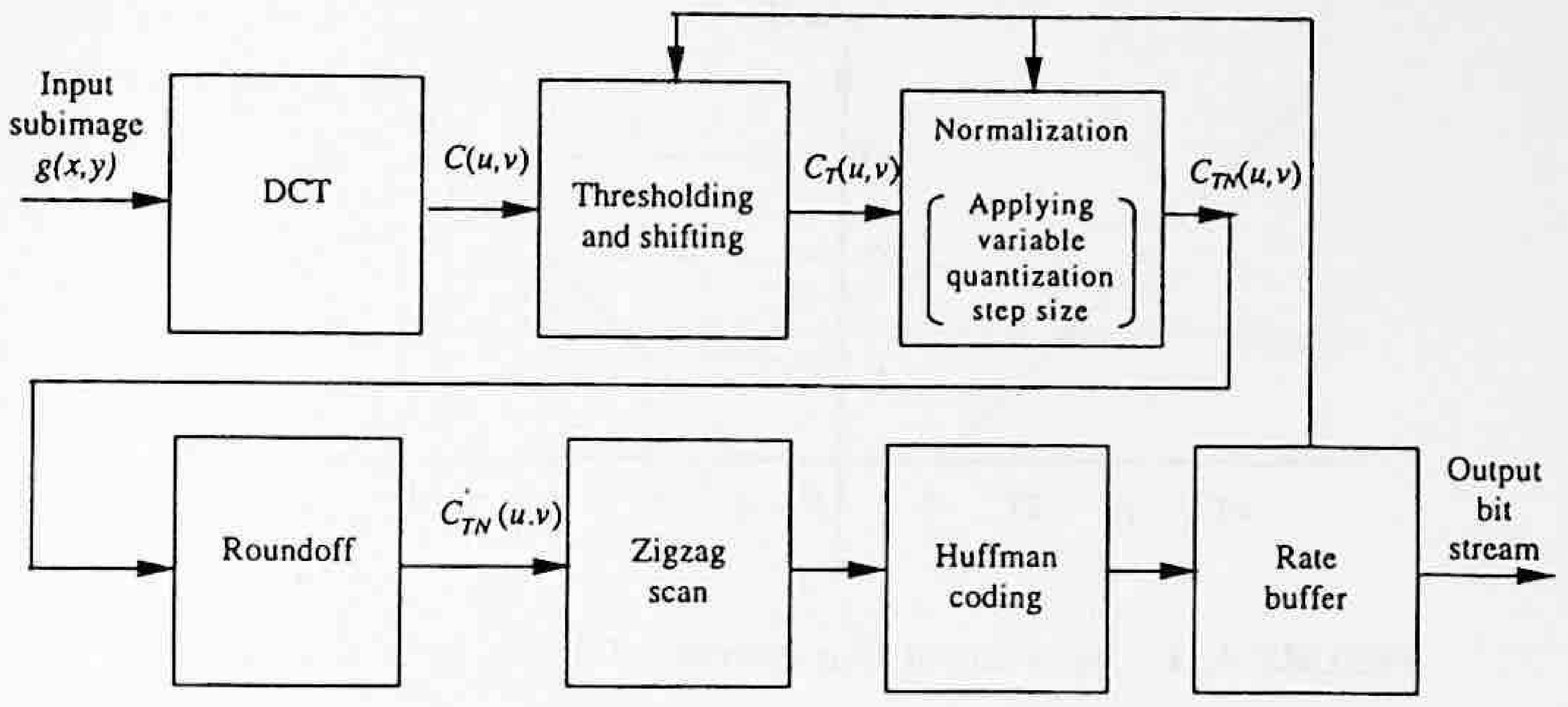
The coding efficiency, however, may not be very high. This is because the zone is predefined based on average statistics. Therefore some coefficients outside the zone might be large in magnitude, while some coefficients inside the zone may be small in quantity. Uniform quantization and natural binary encoding are simple, but they are known not to be efficient enough.

For further improvement of coding efficiency, an adaptive scheme has to be used. There, a two-pass procedure is applied. In the first pass, the variances of transform coefficients are measured or estimated. Based on the statistics, the quantization and encoding schemes are determined. In the second pass, quantization and encoding are carried out (Habibi, 1971a; Chen and Smith, 1977).

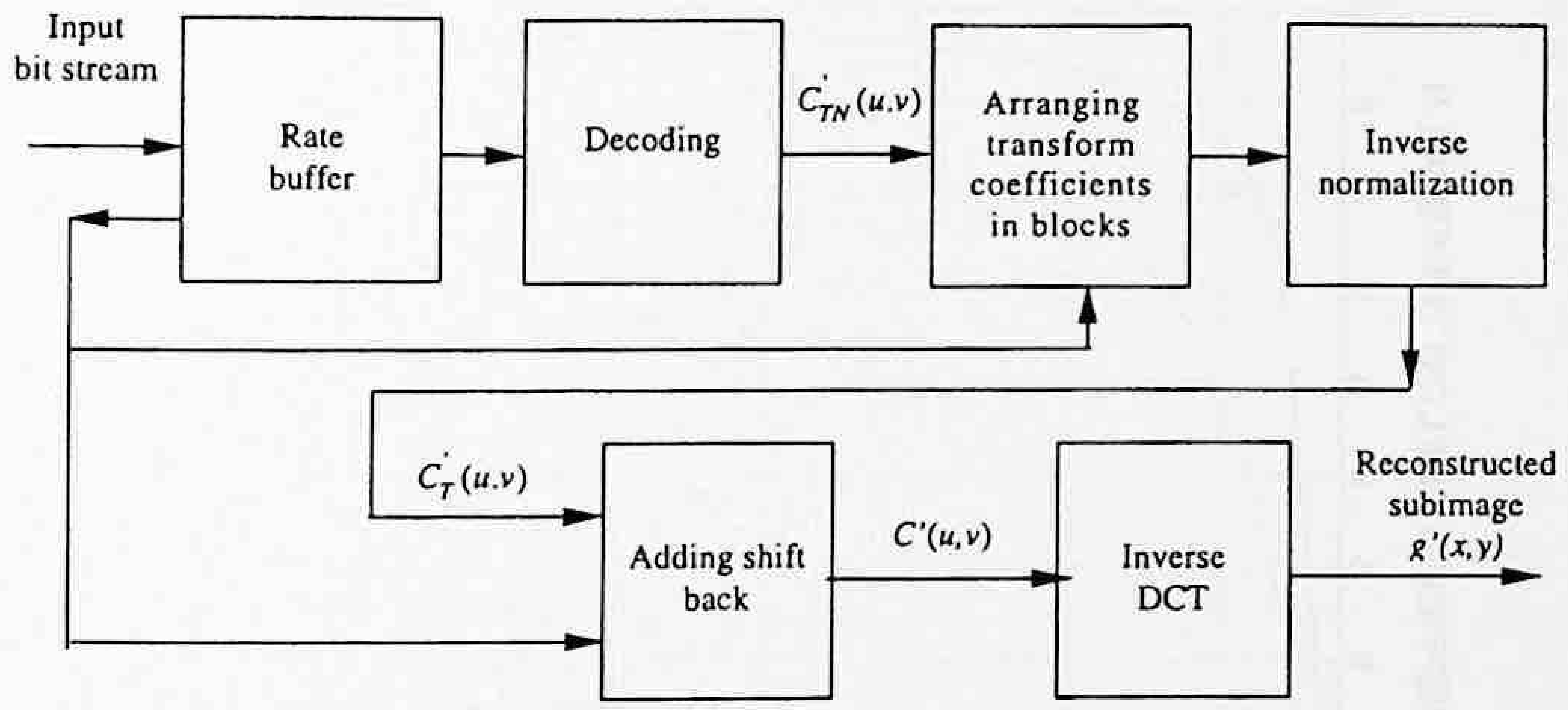
4.4.2 Threshold Coding

In threshold coding, also known as threshold sampling, there is not a predefined zone. Instead, each transform coefficient is compared with a threshold. If it is smaller than the threshold, then it is set to zero. If it is larger than the threshold, it will be retained for quantization and encoding. Compared with zonal coding, this scheme is adaptive in truncation in the sense that the coefficients with more energy are retained no matter where they are located. The addresses of these retained coefficients, however, have to be sent to the receiver as side information. Furthermore, the threshold is determined after an evaluation of all coefficients. Hence, it was usually a two-pass adaptive technique.

Chen and Pratt (1984) devised an efficient adaptive scheme to handle threshold coding. It is a one-pass adaptive scheme, in contrast to the two-pass adaptive schemes. Hence it is fast in implementation. With several effective techniques that will be addressed here, it achieved excellent results in transform coding. Specifically, it demonstrated a satisfactory quality of reconstructed frames at a bit rate of 0.4 bits per pixel for coding of color images, which corresponds to real-time color television transmission over a 1.5-Mb/sec channel. This scheme has been adopted by the international still coding standard JPEG. A block diagram of the threshold coding proposed by Chen and Pratt is shown in Figure 4.10. More details and modification made by JPEG will be described in Chapter 7.



(a) Transmitter



(b) Receiver

FIGURE 4.10 Block diagram of the algorithm proposed by Chen and Pratt (1984).

4.4.2.1 Thresholding and Shifting

The DCT is used in the scheme because of its superiority, described in Section 4.3. Here we use *C(u,v)* to denote the DCT coefficients. The DC coefficient, *C*(0,0), is processed differently. As mentioned in Chapter 3, the DC coefficients are encoded with a differential coding technique. For more details, refer to Chapter 7. For all the AC coefficients, the following thresholding and shifting are carried out:

$$C_T(u,v) = \begin{cases} C(u,v) - T & if \quad C(u,v) > T \\ 0 & if \quad C(u,v) \le T \end{cases} \tag{4.68}$$

where *T* on the right-hand side is the threshold. Note that the above equation also implies a shifting of transform coefficients by *T* when $C(u, v) > T$. The input-output characteristic of the thresholding and shifting is shown in Figure 4.11.

Figure 4.12 demonstrates that more than 60% of the DCT coefficients normally fall below a threshold value as low as 5. This indicates that with a properly selected threshold value it is possible to set most of the DCT coefficients equal to zero. The threshold value is adjusted by the feedback from the rate buffer, or by the desired bit rate.

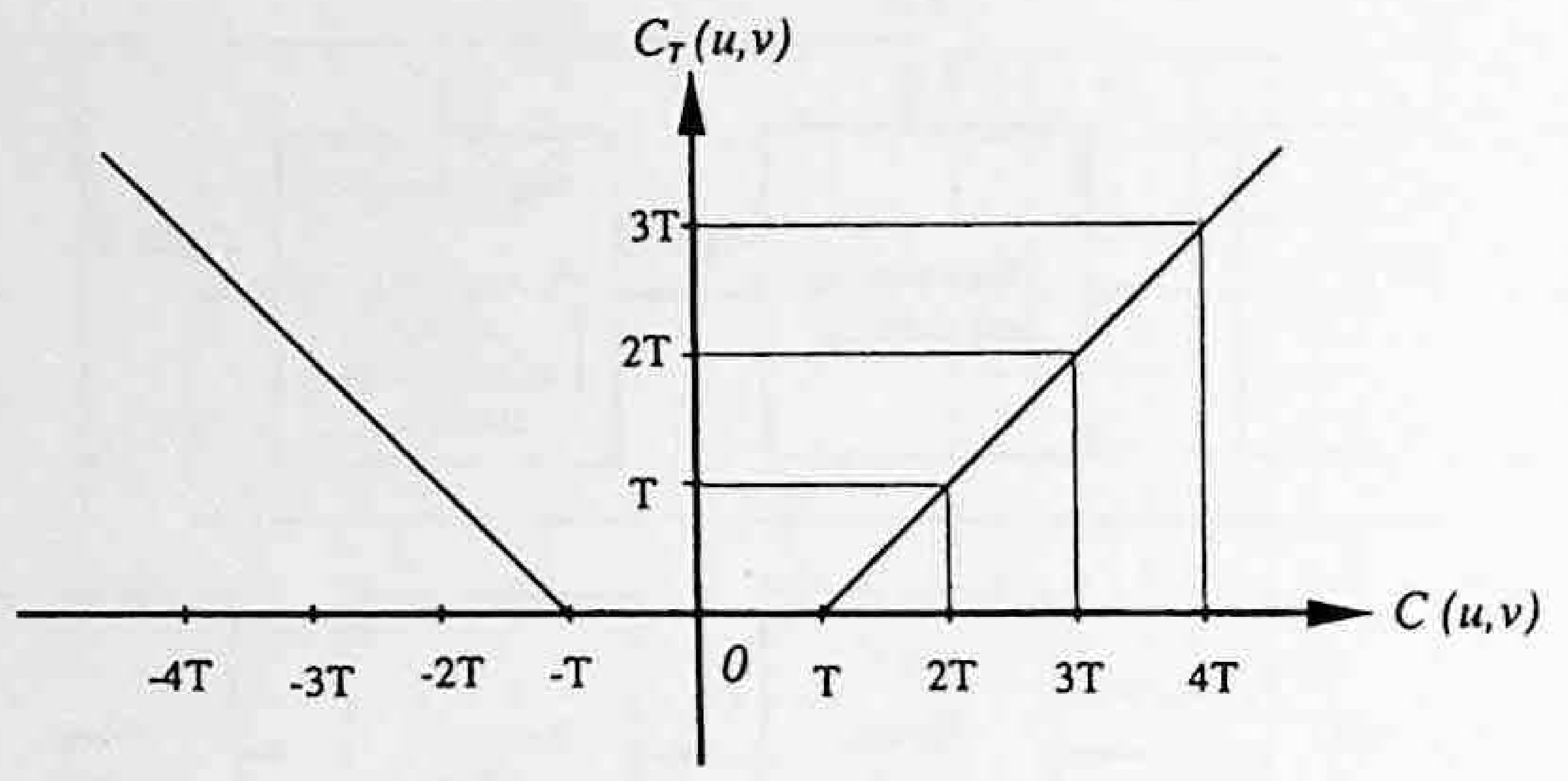


FIGURE 4.11 Input-output characteristic of thresholding and shifting.

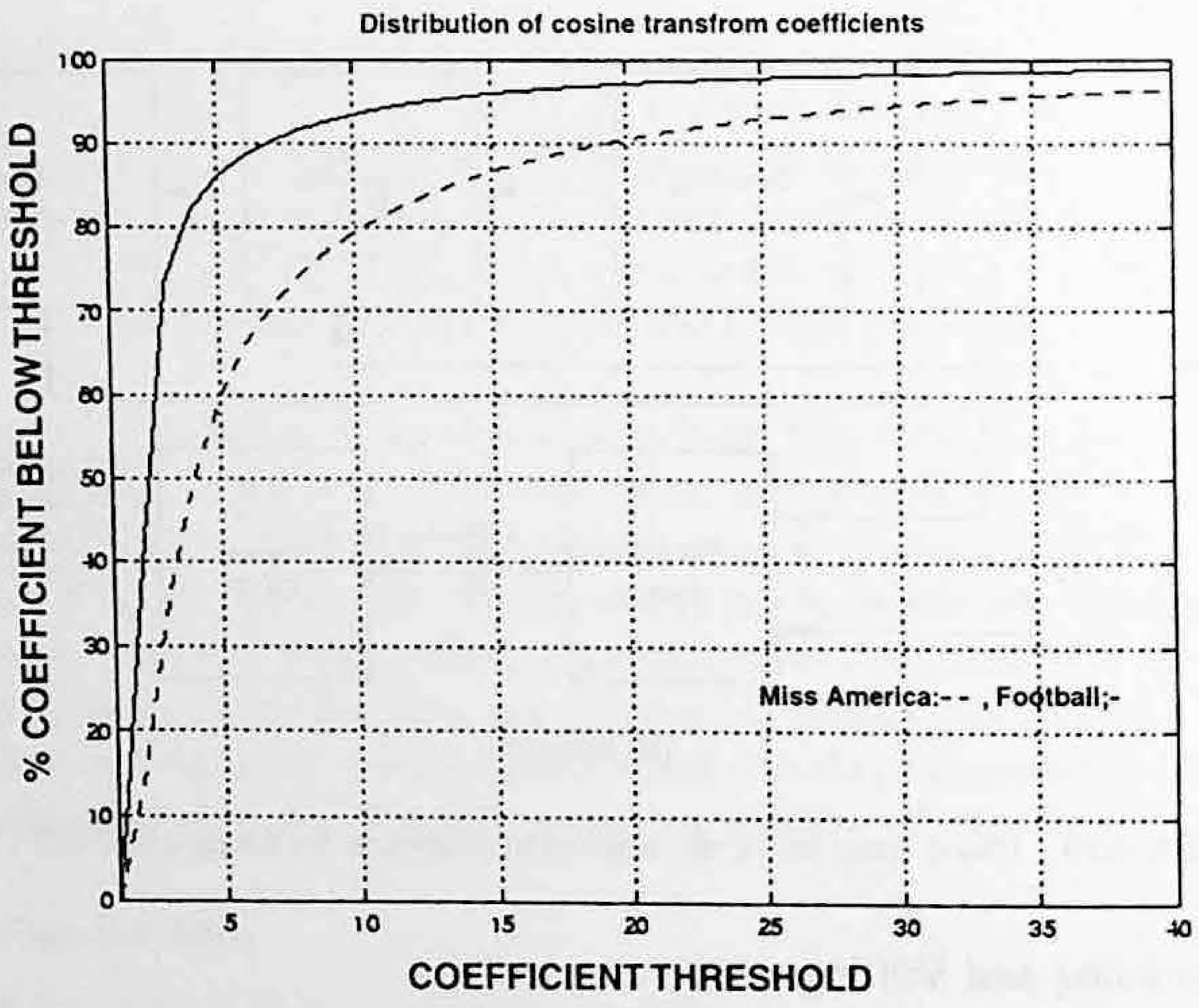


FIGURE 4.12 Amplitude distribution of the DCT coefficients.

4.4.2.2 Normalization and Roundoff

The threshold subtracted transform coefficients $C_T(u,v)$ are normalized before roundoff. The normalization is implemented as follows:

$$C_{TN}(u,v) = \frac{C_T(u,v)}{\Gamma_{u,v}} \tag{4.69}$$

where the normalization factor $\Gamma_{u,v}$ is controlled by the rate buffer. The roundoff process converts floating point to integer as follows.

$$R\left[C_{TN}(u,v)\right] = C_{TN}(u,v) = \begin{cases} \left\lfloor C_{TN}(u,v) + 0.5 \right\rfloor & \text{if} \quad C_{TN}(u,v) \geq 0 \\ \left\lceil C_{TN}(u,v) - 0.5 \right\rceil & \text{if} \quad C_{TN}(u,v) < 0 \end{cases} \tag{4.70}$$

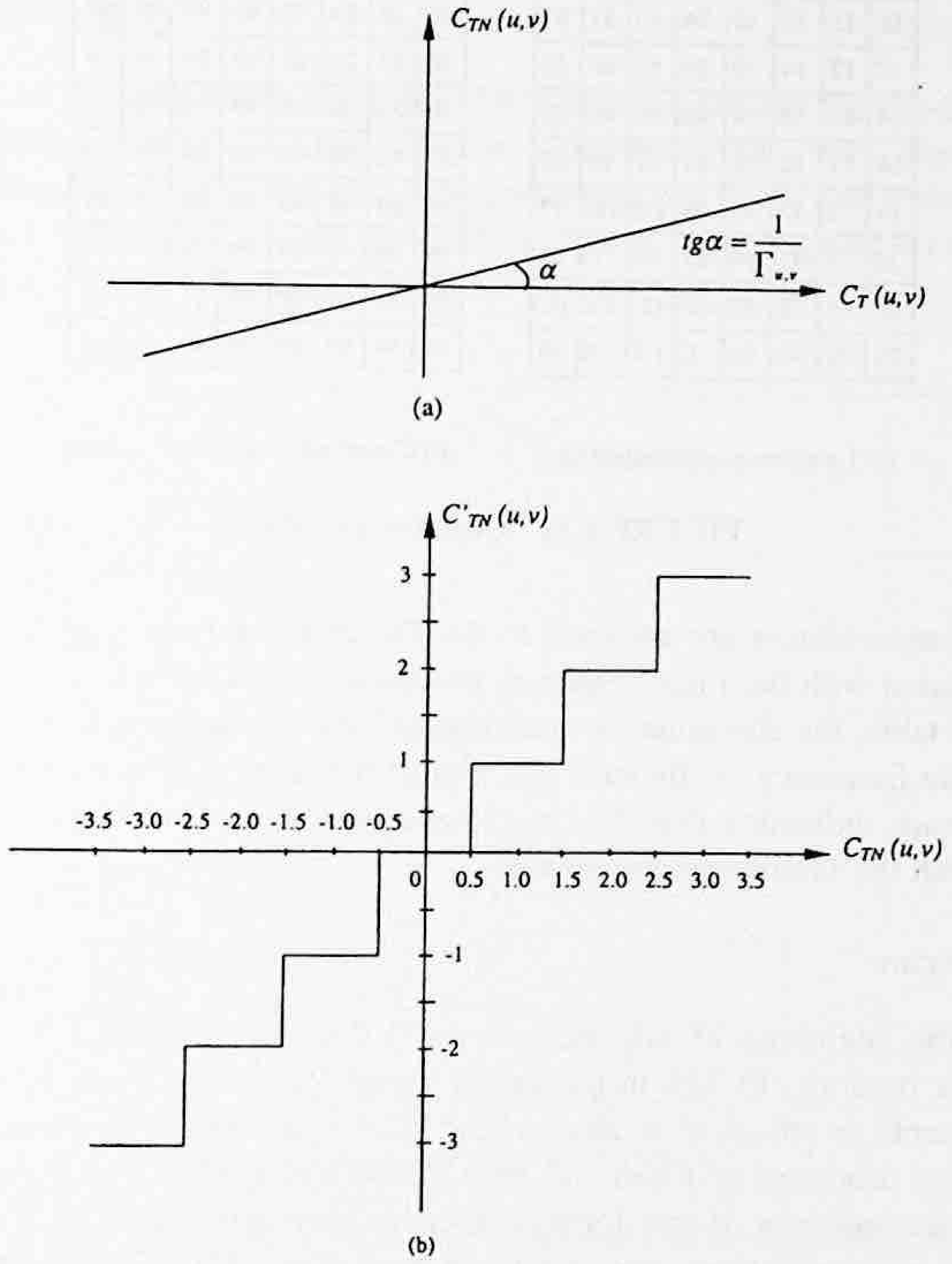


FIGURE 4.13 Input-output characteristic of (a) normalization, (b) roundoff.

where the operator $\lfloor x \rfloor$ means the largest integer smaller than or equal to x, the operator $\lceil x \rceil$ means the smallest integer larger than or equal to x. The input-output characteristics of the normalization and roundoff are shown in Figure 4.13(a) and (b), respectively.

From these input-output characteristics, we can see that the roundoff is a uniform midtread quantizer with a unit quantization step. The combination of normalization and roundoff is equivalent to a uniform midtread quantizer with the quantization step size equal to the normalization factor $\Gamma_{u,v}$. Normalization is a scaling process, which makes the resultant uniform midtread quantizer adapt to the dynamic range of the associated transform coefficient. It is therefore possible for one quantizer design to be applied to various coefficients with different ranges. Obviously, by adjusting the parameter $\Gamma_{u,v}$ (quantization step size) a variable bit rate and mean square quantization error can be achieved. Hence, the selection of the normalization factors for different transform coefficients can take the statistical feature of the images and the characteristics of the human visual system (HVS) into consideration. In general, most image energy is contained in the DC and low-frequency AC transform coefficients. The HVS is more sensitive to a relatively uniform region than to a relatively detailed region, as discussed in Chapter 1. Chapter 1 also mentions that, with regard to the color image, the HVS is more sensitive to the luminance component than to the chrominance components.

These have been taken into consideration in JPEG. A matrix consisting of all the normalization factors is called a quantization table in JPEG. A luminance quantization table and a chrominance quantization table used in JPEG are shown in Figure 4.14. We observe that in general in both tables

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

(a) Luminance quantization table

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

(b) Chrominance quantization table

FIGURE 4.14 Quantization tables.

the small normalization factors are assigned to the DC and low-frequency AC coefficients. The large Γs are associated with the high-frequency transform coefficients. Compared with the luminance quantization table, the chrominance quantization table has larger quantization step sizes for the low- and middle-frequency coefficients and almost the same step sizes for the DC and high-frequency coefficients, indicating that the chrominance components are relatively coarsely quantized, compared with the luminance component.

4.4.2.3 Zigzag Scan

As mentioned at the beginning of this section, while threshold coding is adaptive to the local statistics and hence is more efficient in truncation, threshold coding needs to send the addresses of retained coefficients to the receiver as overhead side information. An efficient scheme, called the zigzag scan, was proposed by Chen and Pratt (1984) and is shown in Figure 4.14. As shown in Figure 4.12, a great majority of transform coefficients have magnitudes smaller than a threshold of 5. Consequently, most quantized coefficients are zero. Hence, in the 1-D sequence obtained by zigzag scanning, most of the numbers are zero. A code known as run-length code, discussed in Chapter 6, is very efficient under these circumstances to encode the address information of nonzero coefficients. Run-length of zero coefficients is understood as the number of consecutive zeros in the zigzag scan. Zigzag scanning minimizes the use of run-length codes in the block.

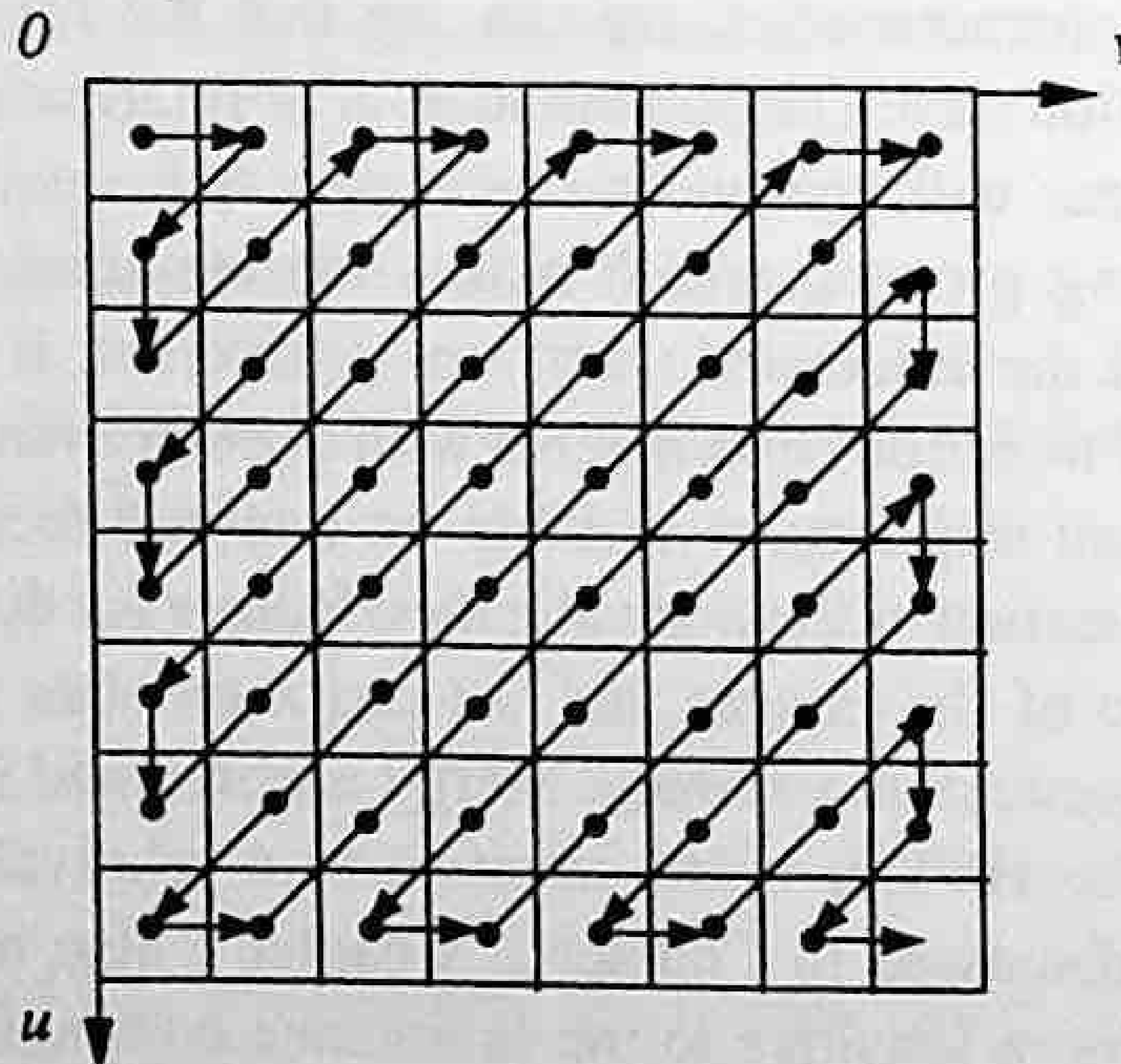


FIGURE 4.15 Zigzag scan of DCT coefficients within an 8 × 8 block.

4.4.2.4 Huffman Coding

Statistical studies of the magnitude of nonzero DCT coefficients and the run-length of zero DCT coefficients in zigzag scanning were conducted in (Chen and Pratt, 1984). The domination of the coefficients with small amplitudes and the short run-lengths was found and is shown in Figures 4.16 and 4.17. This justifies the application of the Huffman coding to the magnitude of nonzero transform coefficients and run-lengths of zeros.

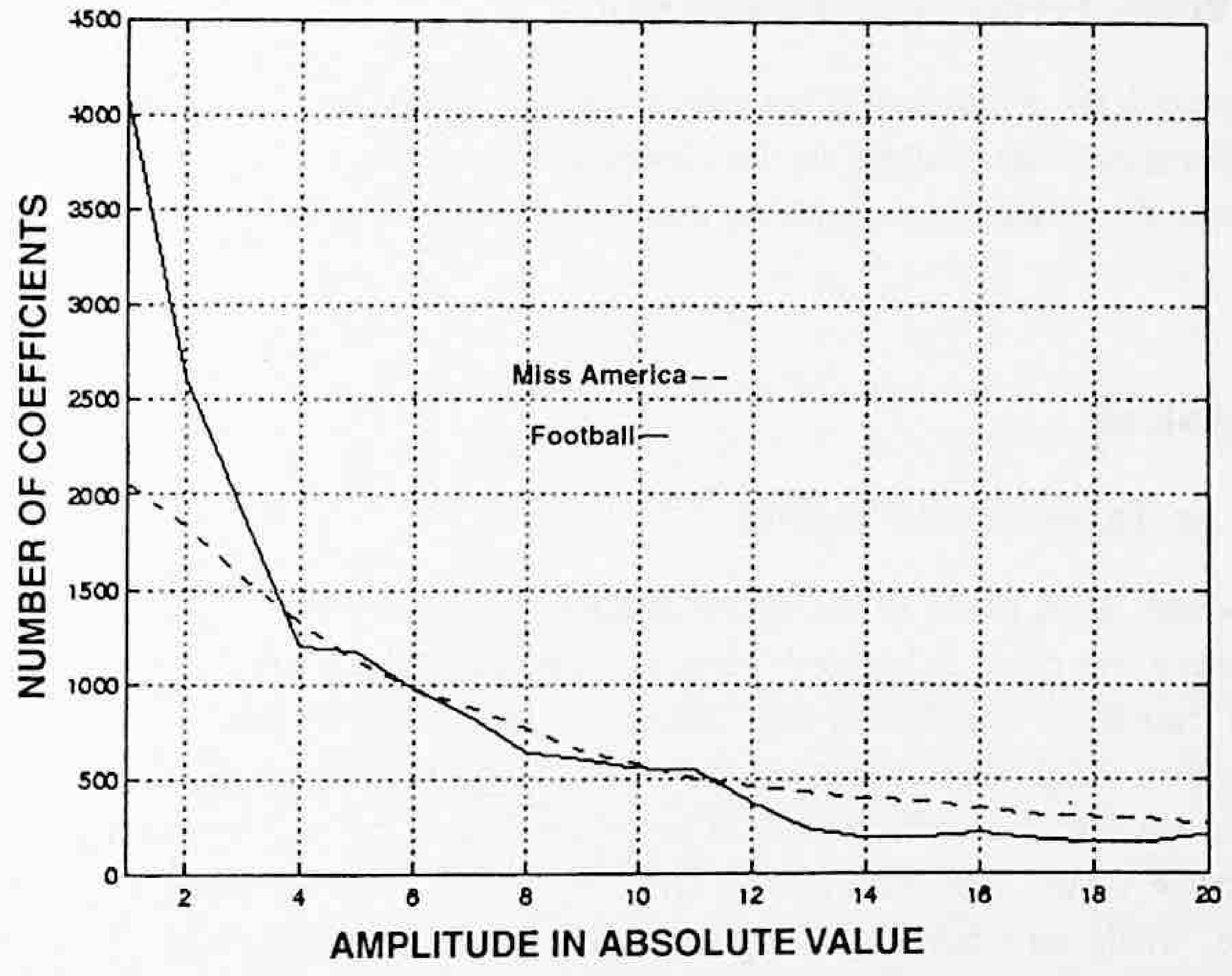


FIGURE 4.16 Histogram of DCT coefficients in absolute amplitude.

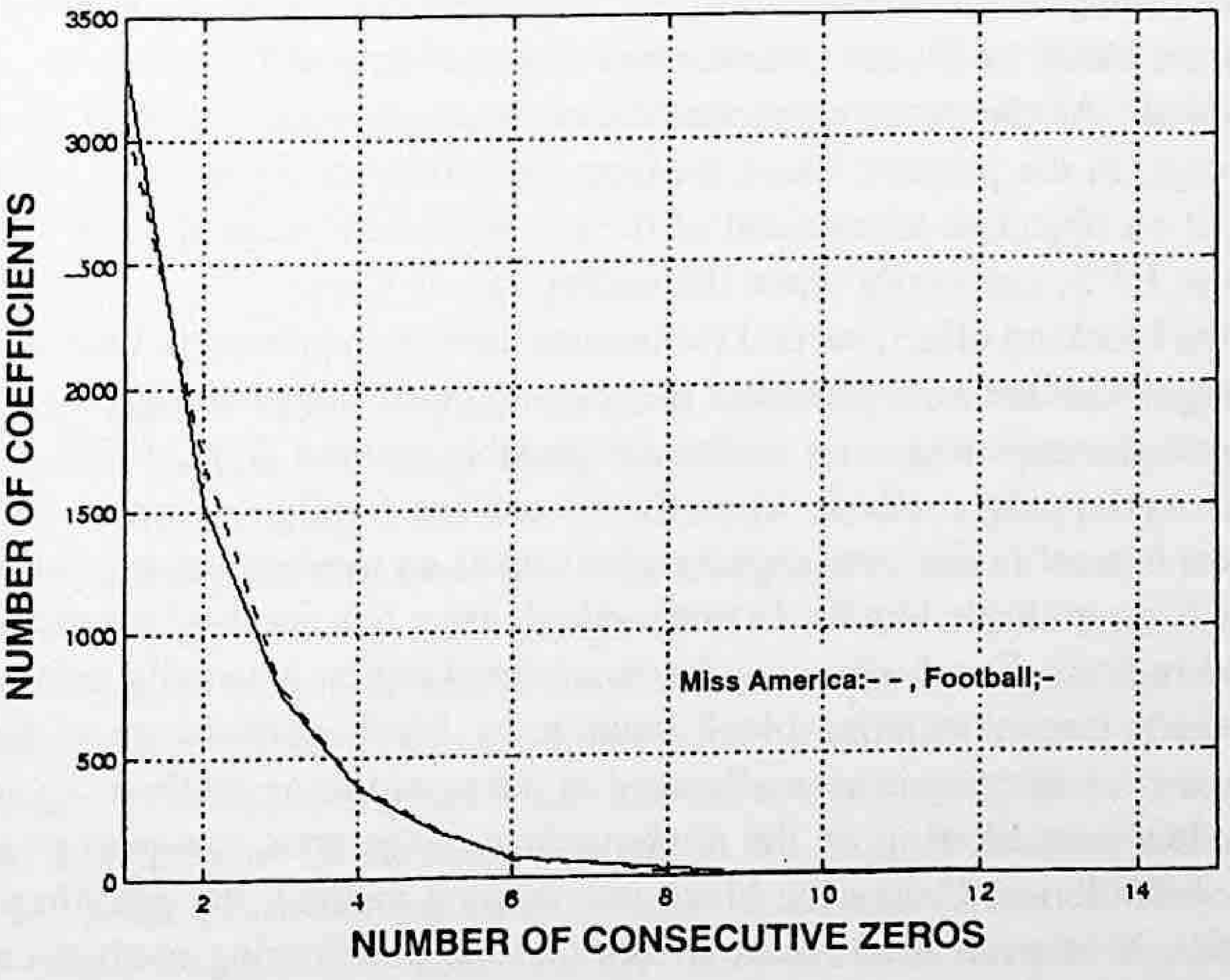


FIGURE 4.17 Histogram of zero run length.

4.4.2.5 Special Codewords

Two special codewords were used by Chen and Pratt (1984). One is called *end of block* (EOB). Another is called *run-length prefix*. Once the last nonzero DCT coefficients in the zigzag is coded, EOB is appended, indicating the termination of coding the block. This further saves bits used in coding. A run-length prefix is used to discriminate the run-length codewords from the amplitude codewords.

4.4.2.6 Rate Buffer Feedback and Equalization

As shown in Figure 4.10, a rate buffer accepts a variable-rate data input from the encoding process and provides a fixed-rate data output to the channel. The status of the rate buffer is monitored and fed back to control the threshold and the normalization factor. In this fashion a one-pass adaptation is achieved.

4.5 SOME ISSUES

4.5.1 Effect of Transmission Errors

In transform coding, each pixel in the reconstructed image relies on all transform coefficients in the subimage where the pixel is located. Hence, a bit reversal transmission error will spread. That is, an error in a transform coefficient will lead to errors in all the pixels within the subimage. As discussed in Section 4.2.3, this is one of the reasons the selected subimage size cannot be very large. Depending on which coefficient is in error, the effect caused by a bit reversal error on the reconstructed image varies. For instance, an error in the DC or a low-frequency AC coefficient may be objectionable, while an error in the high-frequency coefficient may be less noticeable.

4.5.2 Reconstruction Error Sources

As discussed, three sources: truncation (discarding transform coefficients with small variances), quantization, and transmission contribute to the reconstruction error. It is noted that in TC the transform is applied block by block. Quantization and encoding of transform coefficients are also conducted blockwise. At the receiver, reconstructed blocks are put together to form the whole reconstructed image. In the process, block artifacts are produced. Sometimes, even though it may not severely affect an objective assessment of the reconstructed image quality, block artifacts can be annoying to the HVS, especially when the coding rate is low.

To alleviate the blocking effect, several techniques have been proposed. One is to overlap blocks in the source image. Another is to postfilter the reconstructed image along block boundaries. The selection of advanced transforms is an additional possible method (Lim, 1990).

In the block-overlapping method, when the blocks are finally organized to form the reconstructed image, each pixel in the overlapped regions takes an average value of all its reconstructed gray level values from multiple blocks. In this method, extra bits are used for those pixels involved in the overlapped regions. For this reason, the overlapped region is usually only one pixel wide.

Due to the sharp transition along block boundaries, block artifacts are of high frequency in nature. Hence, low-pass filtering is normally used in the postfiltering method. To avoid the blurring effect caused by low-pass filtering on the nonboundary image area, low-pass postfiltering is only applied to block boundaries. Unlike the block-overlapping method, the postfiltering method does not need extra bits. Moreover, it has been shown that the postfiltering method can achieve better results in combating block artifacts (Reeve and Lim, 1984; Ramamurthi and Gersho, 1986). For these two reasons, the postfiltering method has been adopted by the international coding standards.

4.5.3 Comparison Between DPCM and TC

As mentioned at the beginning of the chapter, both differential coding and transform coding utilize interpixel correlation and are efficient coding techniques. Comparisons between these two techniques have been reported (Habibi, 1971b). Take a look at the techniques discussed in the previous chapter and in this chapter. We can see that differential coding is simpler than TC. This is because the linear prediction and differencing involved in differential coding are simpler than the 2-D transform involved in TC. In terms of the memory requirement and processing delay, differential coding such as DPCM is superior to TC. That is, DPCM needs less memory and has less processing delay than TC. The design of the DPCM system, however, is sensitive to image-to-image variation, and so is its performance. That is, an optimum DPCM design is matched to the statistics of a certain image. When the statistics change, the performance of the DPCM will be affected. On the contrary, TC is less sensitive to the variation in the image statistics. In general, the optimum DPCM coding system with a third or higher order predictor performs better than TC when the bit rate is about two to three bits per pixel for single images. When the bit rate is below two to three bits per pixel, TC is normally preferred. As a result, the international still image coding standard JPEG is based on TC, whereas, in JPEG, DPCM is used for coding the DC coefficients of DCT, and information-preserving differential coding is used for lossless still image coding.

4.5.4 Hybrid Coding

A method called hybrid transform/waveform coding, or simply hybrid coding, was devised in order to combine the merits of the two methods. By waveform coding, we mean coding techniques that code the waveform of a signal instead of the transformed signal. DPCM is a waveform coding technique. Hybrid coding combines TC and DPCM coding. That is, TC can be first applied rowwise, followed by DPCM coding columnwise, or vice versa. In this way, the two techniques complement each other. That is, the hybrid coding technique simultaneously has TC's small sensitivity to variable image statistics and DPCM's simplicity in implementation.

Worth mentioning is a successful hybrid coding scheme in interframe coding: predictive coding along the temporal domain. Specifically, it uses motion-compensated predictive coding. That is, the motion analyzed from successive frames is used to more accurately predict a frame. The prediction error (in the 2-D spatial domain) is transform coded. This hybrid coding scheme has been very efficient and was adopted by the international video coding standards H.261, H.263, and MPEG 1, 2, and 4.

4.6 SUMMARY

In transform coding, instead of the original image or some function of the original image in the spatial and/or temporal domain, the image in the transform domain is quantized and encoded. The main idea behind transform coding is that the transformed version of the image is less correlated. Moreover, the image energy is compacted into a small proper subset of transform coefficients.

The basis vector (1-D) and the basis image (2-D) provide a meaningful interpretation of transform coding. This type of interpretation considers the original image to be a weighted sum of basis vectors or basis images. The weights are the transform coefficients, each of which is essentially a correlation measure between the original image and the corresponding basis image. These weights are less correlated than the gray level values of pixels in the original image. Furthermore they have a great disparity in variance distribution. Some weights have large variances. They are retained and finely quantized. Some weights have small energy. They are retained and coarsely quantized. A vast majority of weights are insignificant and discarded. In this way, a high coding efficiency is achieved in transform coding. Because the quantized nonzero coefficients have a very nonuniform probability distribution, they can be encoded by using efficient variable-length codes. In summary, three factors: truncation (discarding a great majority of transform coefficients), adaptive quantization, and variable-length coding contribute mainly to a high coding efficiency of transform coding.

Several linear, reversible, unitary transforms have been studied and utilized in transform coding. They include the discrete Karhunen-Loeve transform (the Hotelling transform), the discrete Fourier transform, the Walsh transform, the Hadamard transform, and the discrete cosine transform. It is shown that the KLT is the optimum. The transform coefficients of the KLT are uncorrelated. The KLT can compact the most energy in the smallest fraction of transform coefficients. However, the KLT is image dependent. There is no fast algorithm to implement it. This prohibits the KLT from practical use in transform coding. While the rest of the transforms perform closely, the DCT appears to be the best. Its energy compaction is very close to the optimum KLT and it can be implemented using the fast Fourier transform. The DCT has been found to be efficient not only for still images coding but also for coding residual images (predictive error) in motion-compensated interframe predictive coding. These features make the DCT the most widely used transform in image and video coding.

There are two ways to truncate transform coefficients: zonal coding and threshold coding. In zonal coding, a zone is predefined based on average statistics. The transform coefficients within the zone are retained, while those outside the zone are discarded. In threshold coding, each transform coefficient is compared with a threshold. Those coefficients larger than the threshold are retained, while those smaller are discarded. Threshold coding is adaptive to local statistics. A two-pass procedure is usually taken. That is, the local statistics are measured or estimated in the first pass. The truncation takes place in the second pass. The addresses of the retained coefficients need to be sent to the receiver as overhead side information.

A one-step adaptive framework of transform coding has evolved as a result of the tremendous research efforts in image coding. It has become a base of the international still image coding standard JPEG. Its fundamental components include the DCT transform, thresholding and adaptive quantization of transform coefficients, zigzag scan, Huffman coding of the magnitude of the nonzero DCT coefficients and run-length of zeros in the zigzag scan, the codeword of EOB, and rate buffer feedback control.

The threshold and the normalization factor are controlled by rate buffer feedback. Since the threshold decides how many transform coefficients are retained and the normalization factor is actually the quantization step size, the rate buffer has direct impact on the bit rate of the transform coding system. Selection of quantization steps takes the energy compaction of the DCT and the characteristics of the HVS into consideration. That is, it uses not only statistical redundancy, but also psychovisual redundancy to enhance coding efficiency.

After thresholding, normalization and roundoff are applied to the DCT transform coefficients in a block; a great majority of transform coefficients are set to zero. A zigzag scan can convert the 2-D array of transform coefficients into a 1-D sequence. The number of consecutive zero-valued coefficients in the 1-D sequence is referred to as the run-length of zeros and is used to provide address information of nonzero DCT coefficients. Both the magnitude of nonzero coefficients and run-length information need to be coded. Statistical analysis has demonstrated that a small magnitude and short run-length are dominant. Therefore, efficient lossless entropy coding methods such as Huffman coding and arithmetic coding (the focus of the next chapter) can be applied to magnitude and run-length.

In a reconstructed subimage, there are three types of errors involved: truncation error (some transform coefficients have been set to zero), quantization error, and transmission error. In a broad sense, the truncation can be viewed as a part of the quantization. That is, these truncated coefficients are quantized to zero. The transmission error in terms of bit reversal will affect the whole reconstructed subimage. This is because, in the inverse transform (such as the inverse DCT), each transform coefficient makes a contribution.

In reconstructing the original image all the subimages are organized to form the whole image. Therefore the independent processing of individual subimages causes block artifacts. Though they may not severely affect the objective assessment of reconstructed image quality, block artifacts can be annoying, especially in low bit rate image coding. Block overlappling and postfiltering are two effective ways to alleviate block artifacts. In the former, neighboring blocks are purposely overlapped by one pixel. In reconstructing the image, those pixels that have been coded more than once take an average of the multiple decoded values. Extra bits are used. In the latter technique, a low-pass filter is applied along boundaries of blocks. No extra bits are required in the process and the effect of combating block artifacts is better than with the former technique.

The selection of subimage size is an important issue in the implementation of transform coding. In general, a large size will remove more interpixel redundancy. But it has been shown that the pixel correlation becomes insignificant when the distance of pixels exceeds 20. On the other hand, a large size is not suitable for adaptation to local statistics, while adaptation is required in handling nonstationary images. A large size also makes the effect of a transmission error spread more widely. For these reasons, subimage size should not be large. In motion-compensated predictive interframe coding, motion estimation is normally carried out in sizes of 16×16 or 8×8. To be compatible, the subimage size in transform coding is normally chosen as 8×8.

Both predictive codings, say, DPCM and TC, utilize interpixel correlation and are efficient coding schemes. Compared with TC, DPCM is simpler in computation. It needs less storage and has less processing delay. But it is more sensitive to image-to-image variation. On the other hand, TC provides higher adaptation to statistical variation. TC is capable of removing more interpixel correlation, thus providing higher coding efficiency. Traditionally, predictive coding is preferred if the bit rate is in the range of two to three bits per pixel, while TC is preferred when bit rate is below two to three bits per pixel. However, the situation changes. TC becomes the core technology in image and video coding. Many special VLSI chips are designed and manufactured for reducing computational complexity. Consequently, predictive coding such as DPCM is only used in some very simple applications.

In the context of interframe coding, 3-D (two spatial dimensions and one temporal dimension) transform coding has not found wide application in practice due to the complexity in computation and storage. Hybrid transform/waveform coding has proven to be very efficient in interframe coding. There, motion-compensated predictive coding is used along the temporal dimension, while transform coding is used to code the prediction error in two spatial dimensions.

4.7 EXERCISES

4-1. Consider the following eight points in a 3-D coordinate system: $(0,0,0)^T$, $(1,0,0)^T$, $(0,1,0)^T$, $(0,0,1)^T$, $(0,1,1)^T$, $(1,0,1)^T$, $(1,1,0)^T$, $(1,1,1)^T$. Find the mean vector and covariance matrix using Equations 4.12 and 4.13.

4-2. For N = 4, find the basis images of the DFT, $I_{u,v}$ when (a) u = 0, v = 0; (b) u = 1, v = 0; (c) u = 2, v = 2; (d) u = 3, v = 2. Use both methods discussed in the text; i.e., the method with basis image and the method with basis vectors.

4-3. For N = 4, find the basis images of the ordered discrete Hadamard transform when (a) u = 0, v = 2; (b) u = 1, v = 3; (c) u = 2, v = 3; (d) u = 3, v = 3. Verify your results by comparing them with Figure 4.5.

4-4. Repeat the previous problem for the DWT, and verify your results by comparing them with Figure 4.4.

4-5. Repeat problem 4-3 for the DCT and N = 4.

4-6. When $N = 8$, draw the transform matrix F for the DWT, DHT, the order DHT, DFT, and DCT.

4-7. The matrix form of forward and inverse 2-D symmetric image transforms are expressed in texts such as (Jayant and Noll, 1984) as $T = FGF^T$ and $G = ITI^T$, which are different from Equations 4.28 and 4.29. Can you explain this discrepancy?
4-8. Derive Equation 4.64. (Hint: use the concept of basis vectors and the orthogonality of basis vectors.)
4-9. Justify that the normalization factor is the quantization step.
4-10. The transform used in TC has two functions: decorrelation and energy compaction. Does decorrelation automatically lead to energy compaction? Comment.
4-11. Using your own words, explain the main idea behind transform coding.
4-12. Read the techniques by Chen and Pratt presented in Section 4.4.2. Compare them with JPEG discussed in Chapter 7. Comment on the similarity and dissimilarity between them.
4-13. How is one-pass adaptation to local statistics in the Chen and Pratt algorithm achieved?
4-14. Explain why the DCT is superior to the DFT in terms of energy compaction.
4-15. Why is the subimage size of 8×8 widely used?

REFERENCES

Ahmed, N., T. Nararajan, and K. R. Rao, Discrete cosine transform, *IEEE Trans. Comput.*, 90-93, 1974.
Andrews, H. C. Multidimensional rotations in feature selection, *IEEE Trans. Comput.*, c-20, 1045-1051, 1971.
Chen, W. H. and C. H. Smith, Adaptive coding of monochrome and color images, *IEEE Trans. Commun.*, COM-25, 1285-1292, 1977.
Chen, W. H. and W. K. Pratt, Scene adaptive coder, *IEEE Trans. Commun.*, COM-32, 225-232, 1984.
Cooley, J. W. and J. W. Tukey, An algorithm for the machine calculation of complex Fourier series, *Math. Comput.*, 19, 297-301, 1965.
Gonzalez, R. C. and R. E. Woods, *Digital Image Processing*, Addison-Wesley, Reading, MA, 1992.
Hadamard, J. Resolution d'une question relative aux determinants, *Bull. Sci. Math., Ser.* 2, 17, Part I, 240-246, 1893.
Habibi, A. and P. A. Wintz, Image coding by linear transformations and block quantization, *IEEE Trans. Commun. Technol.*, com-19, 50-60, 1971.
Habibi, A. Comparison of nth-order DPCM encoder with linear transformations and block quantization techniques, *IEEE Trans. Commun. Technol.*, com-19(6), 948-956, 1971.
Huang, J.-Y. and P. M. Schultheiss, Block quantization of correlated Gaussian random variables, *IEEE Trans. Commun. Syst.*, cs-11, 289-296, 1963.
Jayant, N. S. and P. Noll, *Digital Coding of Waveforms*, Prentice-Hall, Englewood Cliffs, NJ, 1984.
Karhunen, K., On Linear Methods in Probability Theory, Ann. Acad. Sci. Fennicae, Ser. A137. (Translated into English.) The Rand Corp., Santa Monica, CA, 1960.
Lim, J. S. *Two-Dimensional Signal and Image Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1990.
Loève, M., Fonctions Aléatoires de Second Ordre, in P. Levy, *Processus Stochastique et Mouvement Brownien*, Hermann, Paris.
Pearl, J., H. C. Andrews, and W. K. Pratt, Performance measures for transform data coding, *IEEE Trans. Commun. Technol.*, com-20, 411-415, 1972.
Reeve, H. C. III and J. S. Lim, Reduction of blocking effects in image coding, *J. Opt. Eng.*, 23, 34-37, 1984.
Ramamurthi, B. and A. Gersho, Nonlinear space-variant postprocessing of block coded images, *IEEE Trans. Acoust. Speech Signal Process.*, 34, 1258-1267, 1986.
Strang, G., *Introduction to Linear Algebra*, Wellesley-Cambridge Press, 2nd ed., 1998.
Tasto, M. and P. A. Wintz, Image coding by adaptive block quantization, *IEEE Trans. Commun. Technol.*, com-19(6), 957-972, 1971.
Walsh, J. L. A closed set of normal orthogonal functions, *Am. J. Math.*, 45(1), 5-24, 1923.
Wintz, P. A. Transform picture coding, *Proc. IEEE*, 60(7), 809-820, 1972.
Zelinski, R. and P. Noll, Adaptive block quantization of speech signals, (in German), Tech. Rep. no. 181, Heinrich Hertz Institut, Berlin, 1975.

5 Variable-Length Coding: Information Theory Results (II)

Recall the block diagram of encoders shown in Figure 2.3. There are three stages that take place in an encoder: transformation, quantization, and codeword assignment. Quantization was discussed in Chapter 2. Differential coding and transform coding using two different transformation components were covered in Chapters 3 and 4, respectively. In differential coding it is the difference signal that is quantized and encoded, while in transform coding it is the transformed signal that is quantized and encoded. In this chapter and the next chapter, we discuss several codeword assignment (encoding) techniques. In this chapter we cover two types of variable-length coding: Huffman coding and arithmetic coding.

First we introduce some fundamental concepts of encoding. After that, the rules that must be obeyed by all optimum and instantaneous codes are discussed. Based on these rules, the Huffman coding algorithm is presented. A modified version of the Huffman coding algorithm is introduced as an efficient way to dramatically reduce codebook memory while keeping almost the same optimality.

The promising arithmetic coding algorithm, which is quite different from Huffman coding, is another focus of the chapter. While Huffman coding is a *block-oriented* coding technique, arithmetic coding is a *stream-oriented* coding technique. With improvements in implementation, arithmetic coding has gained increasing popularity. Both Huffman coding and arithmetic coding are included in the international still image coding standard JPEG (Joint Photographic [image] Experts Group coding). The adaptive arithmetic coding algorithms have been adopted by the international bilevel image coding standard JBIG (Joint Bi-level Image experts Group coding). Note that the material presented in this chapter can be viewed as a continuation of the information theory results presented in Chapter 1.

5.1 SOME FUNDAMENTAL RESULTS

Prior to presenting Huffman coding and arithmetic coding, we first provide some fundamental concepts and results as necessary background.

5.1.1 Coding an Information Source

Consider an information source, represented by a *source alphabet S*.

$$S = \{s_1, s_2, \cdots, s_m\} \tag{5.1}$$

where s_i, $i = 1,2,\cdots,m$ are *source symbols*. Note that the terms source symbol and information message are used interchangeably in the literature. In this book, however, we would like to distinguish between them. That is, an information message can be a source symbol, or a combination of source symbols. We denote *code alphabet* by A and

$$A = \{a_1, a_2, \cdots, a_r\} \tag{5.2}$$

where a_j, $j = 1,2,\cdots,r$ are *code symbols*. A *message code* is a sequence of code symbols that represents a given information message. In the simplest case, a message consists of only a source symbol. Encoding is then a procedure to assign a *codeword* to the source symbol. Namely,

$$s_i \rightarrow A_i = (a_{i1}, a_{i2}, \cdots, a_{ik}) \tag{5.3}$$

where the codeword A_i is a string of k code symbols assigned to the source symbol s_i. The term message ensemble is defined as the entire set of messages. A code, also known as an ensemble code, is defined as a mapping of all the possible sequences of symbols of S (message ensemble) into the sequences of symbols in A.

Note that in binary coding, the number of code symbols r is equal to 2, since there are only two code symbols available: the binary digits "0" and "1". Two examples are given below to illustrate the above concepts.

Example 5.1
Consider an English article and the ASCII code. Refer to Table 5.1. In this context, the source alphabet consists of all the English letters in both lower and upper cases and all the punctuation marks. The code alphabet consists of the binary 1 and 0. There are a total of 128 7-bit binary codewords. From Table 5.1, we see that the codeword assigned to the capital letter A is 1000001. That is, A is a source symbol, while 1000001 is its codeword.

Example 5.2
Table 5.2 lists what is known as the (5,2) code. It is a linear block code. In this example, the source alphabet consists of the four (2^2) source symbols listed in the left column of the table: 00, 01, 10, and 11. The code alphabet consists of the binary 1 and 0. There are four codewords listed in the right column of the table. From the table, we see that the code assigns a 5-bit codeword to each source symbol. Specifically, the codeword of the source symbol 00 is 00000. The source symbol 01 is encoded as 10100; 01111 is the codeword assigned to 10. The symbol 11 is mapped to 11011.

5.1.2 Some Desired Characteristics

To be practical in use, codes need to have some desirable characteristics (Abramson, 1963). Some of the characteristics are addressed in this subsection.

5.1.2.1 Block Code

A code is said to be a block code if it maps each source symbol in S into a fixed codeword in A. Hence, the codes listed in the above two examples are block codes.

5.1.2.2 Uniquely Decodable Code

A code is uniquely decodable if it can be unambiguously decoded. Obviously, a code has to be uniquely decodable if it is to be of use.

Example 5.3
Table 5.3 specifies a code. Obviously it is not uniquely decodable since if a binary string "00" is received we do not know which of the following two source symbols has been sent out: s_1 or s_3.

Nonsingular Code
A block code is nonsingular if all the codewords are distinct (see Table 5.4).

Example 5.4
Table 5.4 gives a nonsingular code since all four codewords are distinct. If a code is not a nonsingular code, i.e., at least two codewords are identical, then the code is not uniquely decodable. Notice, however, that a nonsingular code does not guarantee unique decodability. The code shown in Table 5.4 is such an example in that it is nonsingular while it is not uniquely decodable. It is not

TABLE 5.1
Seven-Bit American Standard Code for Information Interchange (ASCII)

Bits												
				5	0	1	0	1	0	1	0	1
				6	0	0	1	1	0	0	1	1
1	2	3	4	7	0	0	0	0	1	1	1	1
0	0	0	0		NUL	DLE	SP	0	@	P	‘	p
1	0	0	0		SOH	DC1	!	1	A	Q	a	q
0	1	0	0		STX	DC2	″	2	B	R	b	r
1	1	0	0		ETX	DC3	#	3	C	S	c	s
0	0	1	0		EOT	DC4	$	4	D	T	d	t
1	0	1	0		ENQ	NAK	%	5	E	U	e	u
0	1	1	0		ACK	SYN	&	6	F	V	f	v
1	1	1	0		BEL	ETB	′	7	G	W	g	w
0	0	0	1		BS	CAN	(	8	H	X	h	x
1	0	0	1		HT	EM	)	9	I	Y	i	y
0	1	0	1		LF	SUB	*	:	J	Z	j	z
1	1	0	1		VT	ESC	+	;	K	[	k	{
0	0	1	1		FF	FS	,	<	L	\	l	|
1	0	1	1		CR	GS	-	=	M	]	m	}
0	1	1	1		SO	RS	.	>	N	∧	n	~
1	1	1	1		SI	US	/	?	O	—	o	DEL

NUL	Null, or all zeros	DC1	Device control 1
SOH	Start of heading	DC2	Device control 2
STX	Start of text	DC3	Device control 3
ETX	End of text	DC4	Device control 4
EOT	End of transmission	NAK	Negative acknowledgment
ENQ	Enquiry	SYN	Synchronous idle
ACK	Acknowledge	ETB	End of transmission block
BEL	Bell, or alarm	CAN	Cancel
BS	Backspace	EM	End of medium
HT	Horizontal tabulation	SUB	Substitution
LF	Line feed	ESC	Escape
VT	Vertical tabulation	FS	File separator
FF	Form feed	GS	Group separator
CR	Carriage return	RS	Record separator
SO	Shift out	US	Unit separator
SI	Shift in	SP	Space
DLE	Data link escape	DEL	Delete

TABLE 5.2
A (5,2) Linear Block Code

Source Symbol	Codeword
S_1 (0 0)	0 0 0 0 0
S_2 (0 1)	1 0 1 0 0
S_3 (1 0)	0 1 1 1 1
S_4 (1 1)	1 1 0 1 1

TABLE 5.3
A Not Uniquely Decodable Code

Source Symbol	Codeword
S_1	0 0
S_2	1 0
S_3	0 0
S_4	1 1

TABLE 5.4
A Nonsingular Code

Source Symbol	Codeword
S_1	1
S_2	1 1
S_3	0 0
S_4	0 1

uniquely decodable because once the binary string "11" is received, we do not know if the source symbols transmitted are s_1 followed by s_1 or simply s_2.

The *n*th Extension of a Block Code

The nth extension of a block code, which maps the source symbol s_i into the codeword A_i, is a block code that maps the sequences of source symbols $s_{i1}s_{i2} \cdots s_{in}$ into the sequences of codewords $A_{i1}A_{i2} \cdots A_{in}$.

A Necessary and Sufficient Condition of a Block Code's Unique Decodability

A block code is uniquely decodable if and only if the nth extension of the code is nonsingular for every finite n.

Example 5.5

The second extension of the nonsingular block code shown in Example 5.4 is listed in Table 5.5. Clearly, this second extension of the code is not a nonsingular code, since the entries s_1s_2 and s_2s_1 are the same. This confirms the nonunique decodability of the nonsingular code in Example 5.4.

TABLE 5.5
The Second Extension of the Nonsingular Block Code in Example 5.4

Source Symbol	Codeword	Source Symbol	Codeword
$S_1 S_1$	1 1	$S_3 S_1$	0 0 1
$S_1 S_2$	1 1 1	$S_3 S_2$	0 0 1 1
$S_1 S_3$	1 0 0	$S_3 S_3$	0 0 0 0
$S_1 S_4$	1 0 1	$S_3 S_4$	0 0 0 1
$S_2 S_1$	1 1 1	$S_4 S_1$	0 1 1
$S_2 S_2$	1 1 1 1	$S_4 S_2$	0 1 1 1
$S_2 S_3$	1 1 0 0	$S_4 S_3$	0 1 0 0
$S_2 S_4$	1 1 0 1	$S_4 S_4$	0 1 0 1

TABLE 5.6
Three Uniquely Decodable Codes

Source Symbol	Code 1	Code 2	Code 3
S_1	0 0	1	1
S_2	0 1	0 1	10
S_3	1 0	0 0 1	1 0 0
S_4	1 1	0 0 0 1	1 0 0 0

5.1.2.3 Instantaneous Codes

Definition of Instantaneous Codes
A uniquely decodable code is said to be instantaneous if it is possible to decode each codeword in a code symbol sequence without knowing the succeeding codewords.

Example 5.6
Table 5.6 lists three uniquely decodable codes. The first one is in fact a two-bit natural binary code. In decoding, we can immediately tell which source symbols are transmitted since each codeword has the same length. In the second code, code symbol "1" functions like a comma. Whenever we see a "1", we know it is the end of the codeword. The third code is different from the previous two codes in that if we see a "10" string we are not sure if it corresponds to s_2 until we see a succeeding "1". Specifically, if the next code symbol is "0", we still cannot tell if it is s_3 since the next one may be "0" (hence s_4) or "1" (hence s_3). In this example, the next "1" belongs to the succeeding codeword. Therefore we see that code 3 is uniquely decodable. It is not instantaneous, however.

Definition of the *j*th Prefix
Assume a codeword $A_i = a_{i1}a_{i2}\cdots a_{ik}$. Then the sequences of code symbols $a_{i1}a_{i2}\cdots a_{ij}$ with $1 \leq j \leq k$ is the jth order prefix of the codeword A_i.

Example 5.7
If a codeword is 11001, it has the following five prefixes: 11001, 1100, 110, 11, 1. The first-order prefix is 1, while the fifth-order prefix is 11001.

A Necessary and Sufficient Condition of Being an Instantaneous Code
A code is instantaneous if and only if no codeword is a prefix of some other codeword. This condition is often referred to as the *prefix condition*. Hence, the instantaneous code is also called the prefix condition code or sometimes simply the prefix code. In many applications, we need a block code that is nonsingular, uniquely decodable, and instantaneous.

5.1.2.4 Compact Code

A uniquely decodable code is said to be compact if its average length is the minimum among all other uniquely decodable codes based on the same source alphabet S and code alphabet A. A compact code is also referred to as a *minimum redundancy* code, or an *optimum* code.

Note that the average length of a code was defined in Chapter 1 and is restated below.

5.1.3 Discrete Memoryless Sources

This is the simplest model of an information source. In this model, the symbols generated by the source are independent of each other. That is, the source is memoryless or it has a zero memory.

Consider the information source expressed in Equation 5.1 as a discrete memoryless source. The occurrence probabilities of the source symbols can be denoted by $p(s_1)$, $p(s_2)$, $\cdots$, $p(s_m)$. The lengths of the codewords can be denoted by $l_1, l_2, \cdots, l_m$. The average length of the code is then equal to

$$L_{avg} = \sum_{i=1}^{m} l_i p(s_i) \tag{5.4}$$

Recall Shannon's first theorem, i.e., the noiseless coding theorem described in Chapter 1. The average length of the code is bounded below by the entropy of the information source. The entropy of the source S is defined as $H(S)$ and

$$H(S) = -\sum_{i=1}^{m} p(s_i) \log_2 p(s_i) \tag{5.5}$$

Recall that entropy is the average amount of information contained in a source symbol. In Chapter 1 the efficiency of a code, η, is defined as the ratio between the entropy and the average length of the code. That is, $\eta = H(S)/L_{avg}$. The redundancy of the code, ζ, is defined as $\zeta = 1 - \eta$.

5.1.4 Extensions of a Discrete Memoryless Source

Instead of coding each source symbol in a discrete source alphabet, it is often useful to code blocks of symbols. It is, therefore, necessary to define the nth extension of a discrete memoryless source.

5.1.4.1 Definition

Consider the zero-memory source alphabet S defined in Equation 5.1. That is, $S = \{s_1, s_2, \cdots, s_m\}$. If n symbols are grouped into a block, then there is a total of m^n blocks. Each block is considered as a new source symbol. These m^n blocks thus form an information source alphabet, called the nth extension of the source S, which is denoted by S^n.

5.1.4.2 Entropy

Let each block be denoted by β_i and

$$\beta_i = (s_{i1}, s_{i2}, \cdots, s_{in}) \tag{5.6}$$

Then we have the following relation due to the memoryless assumption:

$$p(\beta_i) = \prod_{j=1}^{n} p(s_{ij}) \tag{5.7}$$

Hence, the relationship between the entropy of the source S and the entropy of its nth extension is as follows:

$$H(S^n) = n \cdot H(S) \tag{5.8}$$

Example 5.8
Table 5.7 lists a source alphabet. Its second extension is listed in Table 5.8.

TABLE 5.7
A Discrete Memoryless Source Alphabet

Source Symbol	Occurrence Probability
S_1	0.6
S_2	0.4

TABLE 5.8
The Second Extension of the Source Alphabet Shown in Table 5.7

Source Symbol	Occurrence Probability
$S_1 S_1$	0.36
$S_2 S_2$	0.24
$S_2 S_1$	0.24
$S_2 S_2$	0.16

The entropy of the source and its second extension are calculated below.

$$H(S) = -0.6 \cdot \log_2(0.6) - 0.4 \cdot \log_2(0.4) \approx 0.97$$

$$H(S^2) = -0.36 \cdot \log_2(0.36) - 2 \cdot 0.24 \cdot \log_2(0.24) - 0.16 \cdot \log_2(0.16) \approx 1.94$$

It is seen that $H(S^2) = 2H(S)$.

5.1.4.3 Noiseless Source Coding Theorem

The noiseless source coding theorem, also known as Shannon's first theorem, defining the minimum average codeword length per source pixel, was presented in Chapter 1, but without a mathematical expression. Here, we provide some mathematical expressions in order to give more insight about the theorem.

For a discrete zero-memory information source S, the noiseless coding theorem can be expressed as

$$H(S) \le L_{avg} < H(S) + 1 \tag{5.9}$$

That is, there exists a variable-length code whose average length is bounded below by the entropy of the source (that is encoded) and bounded above by the entropy plus 1. Since the nth extension of the source alphabet, S^n, is itself a discrete memoryless source, we can apply the above result to it. That is,

$$H(S^n) \le L^n_{avg} < H(S^n) + 1 \tag{5.10}$$

where L^n_{avg} is the average codeword length of a variable-length code for the S^n. Since $H(S^n) = nH(S)$ and $L^n_{avg} = nL^n$avg, we have

$$H(S) \le L_{avg} < H(S) + \frac{1}{n} \tag{5.11}$$

Therefore, when coding blocks of n source symbols, the noiseless source coding theory states that for an arbitrary positive number ε, there is a variable-length code which satisfies the following:

$$H(S) \le L_{avg} < H(S) + \varepsilon \tag{5.12}$$

as n is large enough. That is, the average number of bits used in coding per source symbol is bounded below by the entropy of the source and is bounded above by the sum of the entropy and an arbitrary positive number. To make ε arbitrarily small, i.e., to make the average length of the code arbitrarily close to the entropy, we have to make the block size n large enough. This version of the noiseless coding theorem suggests a way to make the average length of a variable-length code approach the source entropy. It is known, however, that the high coding complexity that occurs when n approaches infinity makes implementation of the code impractical.

5.2 HUFFMAN CODES

Consider the source alphabet defined in Equation 5.1. The method of encoding source symbols according to their probabilities, suggested in (Shannon, 1948; Fano, 1949), is not optimum. It approaches the optimum, however, when the block size n approaches infinity. This results in a large storage requirement and high computational complexity. In many cases, we need a direct encoding method that is optimum and instantaneous (hence uniquely decodable) for an information source with finite source symbols in source alphabet S. Huffman code is the first such optimum code (Huffman, 1952), and is the technique most frequently used at present. It can be used for r-ary encoding as $r > 2$. For notational brevity, however, we discuss only the Huffman coding used in the binary case presented here.

5.2.1 Required Rules for Optimum Instantaneous Codes

Let us rewrite Equation 5.1 as follows:

$$S = (s_1, s_2, \cdots, s_m) \tag{5.13}$$

Without loss of generality, assume the occurrence probabilities of the source symbols are as follows:

$$p(s_1) \ge p(s_2) \ge \cdots \ge p(s_{m-1}) \ge p(s_m) \tag{5.14}$$

Since we are seeking the optimum code for S, the lengths of codewords assigned to the source symbols should be

$$l_1 \le l_2 \le \cdots \le l_{m-1} \le l_m . \tag{5.15}$$

Based on the requirements of the optimum and instantaneous code, Huffman derived the following rules (restrictions):

1. $l_1 \le l_2 \le \cdots \le l_{m-1} = l_m$ (5.16)

   Equations 5.14 and 5.16 imply that when the source symbol occurrence probabilities are arranged in a nonincreasing order, the length of the corresponding codewords should be in a nondecreasing order. In other words, the codeword length of a more probable source symbol should not be longer than that of a less probable source symbol. Furthermore, the length of the codewords assigned to the two least probable source symbols should be the same.

2. The codewords of the two least probable source symbols should be the same except for their last bits.
3. Each possible sequence of length $l_m - 1$ bits must be used either as a codeword or must have one of its prefixes used as a codeword.

Rule 1 can be justified as follows. If the first part of the rule, i.e., $l_1 \le l_2 \le \cdots \le l_{m-1}$ is violated, say, $l_1 > l_2$, then we can exchange the two codewords to shorten the average length of the code. This means the code is not optimum, which contradicts the assumption that the code is optimum. Hence it is impossible. That is, the first part of Rule 1 has to be the case. Now assume that the second part of the rule is violated, i.e., $l_{m-1} < l_m$. (Note that $l_{m-1} > l_m$ can be shown to be impossible by using the same reasoning we just used in proving the first part of the rule.) Since the code is instantaneous, codeword A_{m-1} is not a prefix of codeword A_m. This implies that the last bit in the codeword A_m is redundant. It can be removed to reduce the average length of the code, implying that the code is not optimum. This contradicts the assumption, thus proving Rule 1.

Rule 2 can be justified as follows. As in the above, A_{m-1} and A_m are the codewords of the two least probable source symbols. Assume that they do not have the identical prefix of the order $l_m - 1$. Since the code is optimum and instantaneous, codewords A_{m-1} and A_m cannot have prefixes of any order that are identical to other codewords. This implies that we can drop the last bits of A_{m-1} and A_m to achieve a lower average length. This contradicts the optimum code assumption. It proves that Rule 2 has to be the case.

Rule 3 can be justified using a similar strategy to that used above. If a possible sequence of length $l_m - 1$ has not been used as a codeword and any of its prefixes have not been used as codewords, then it can be used in place of the codeword of the *m*th source symbol, resulting in a reduction of the average length L_{avg}. This is a contradiction to the optimum code assumption and it justifies the rule.

5.2.2 Huffman Coding Algorithm

Based on these three rules, we see that the two least probable source symbols have codewords of equal length. These two codewords are identical except for the last bits, the binary 0 and 1, respectively. Therefore, these two source symbols can be combined to form a single new symbol. Its occurrence probability is the sum of two source symbols, i.e., $p(s_{m-1}) + p(s_m)$. Its codeword is the common prefix of order $l_m - 1$ of the two codewords assigned to s_m and s_{m-1}, respectively. The new set of source symbols thus generated is referred to as the first auxiliary source alphabet, which is one source symbol less than the original source alphabet. In the first auxiliary source alphabet, we can rearrange the source symbols according to a nonincreasing order of their occurrence probabilities. The same procedure can be applied to this newly created source alphabet. A binary 0 and a binary 1, respectively, are assigned to the last bits of the two least probable source symbols in the alphabet. The second auxiliary source alphabet will again have one source symbol less than the first auxiliary source alphabet. The procedure continues. In some step, the resultant source alphabet will have only two source symbols. At this time, we combine them to form a single source symbol with a probability of 1. The coding is then complete.

Let's go through the following example to illustrate the above Huffman algorithm.

Example 5.9
Consider a source alphabet whose six source symbols and their occurrence probabilities are listed in Table 5.9. Figure 5.1 demonstrates the Huffman coding procedure applied. In the example, among the two least probable source symbols encountered at each step we assign binary 0 to the top symbol and binary 1 to the bottom symbol.