9 Nonstandard Image Coding

In this chapter, we introduce three nonstandard image coding techniques: vector quantization (VQ) (Nasrabadi and King, 1988), fractal coding (Barnsley and Hurd, 1993; Fisher, 1994; Jacquin, 1993), and model-based coding (Li et al., 1994).

9.1 INTRODUCTION

The VQ, fractal coding, and model-based coding techniques have not yet been adopted as an image coding standard. However, due to their unique features these techniques may find some special applications. Vector quantization is an effective technique for performing data compression. Theoretically, vector quantization is always better than scalar quantization because it fully exploits the correlation between components within the vector. The optimal coding performance will be obtained when the dimension of the vector approaches infinity, and then the correlation between all components is exploited for compression. Another very attractive feature of image vector quantization is that its decoding procedure is very simple since it only consists of table look-ups. However, there are two major problems with image VQ techniques. The first is that the complexity of vector quantization exponentially increases with the increasing dimensionality of vectors. Therefore, for vector quantization it is important to solve the problem of how to design a practical coding system which can provide a reasonable performance under a given complexity constraint. The second major problem of image VQ is the need for a codebook, which causes several problems in practical application such as generating a universal codebook for a large number of images, scaling the codebook to fit the bit rate requirement, and so on. Recently, the lattice VQ schemes have been proposed to address these problems (Li, 1997).

Fractal theory has a long history. Fractal-based techniques have been used in several areas of digital image processing such as image segmentation, image synthesis, and computer graphics, but only in recent years have they been extended to the applications of image compression (Jacquin, 1993). A fractal is a geometric form which has the unique feature of having extremely high visual self-similar irregular details while containing very low information content. Several methods for image compression have been developed based on different characteristics of fractals. One method is based on Iterated Function Systems (*IFS*) proposed by Barnsley (1988). This method uses the self-similar and self-affine property of fractals. Such a system consists of sets of transformations including translation, rotation, and scaling. On the encoder side of a fractal image coding system, a set of fractals is generated from the input image. These fractals can be used to reconstruct the image at the decoder side. Since these fractals are represented by very compact fractal transformations, they require very small amounts of data to be expressed and stored as formulas. Therefore, the information needed to be transmitted is very small. The second fractal image coding method is based on the fractal dimension (Lu, 1993; Jang and Rajala, 1990). Fractal dimension is a good representation of the roughness of image surfaces. In this method, the image is first segmented using the fractal dimension and then the resultant uniform segments can be efficiently coded using the properties of the human visual system. Another fractal image coding scheme is based on fractal geometry, which is used to measure the length of a curve with a yardstick (Walach, 1989). The details of these coding methods will be discussed in Section 9.3.

The basic idea of model-based coding is to reconstruct an image with a set of model parameters. The model parameters are then encoded and transmitted to the decoder. At the decoder the decoded

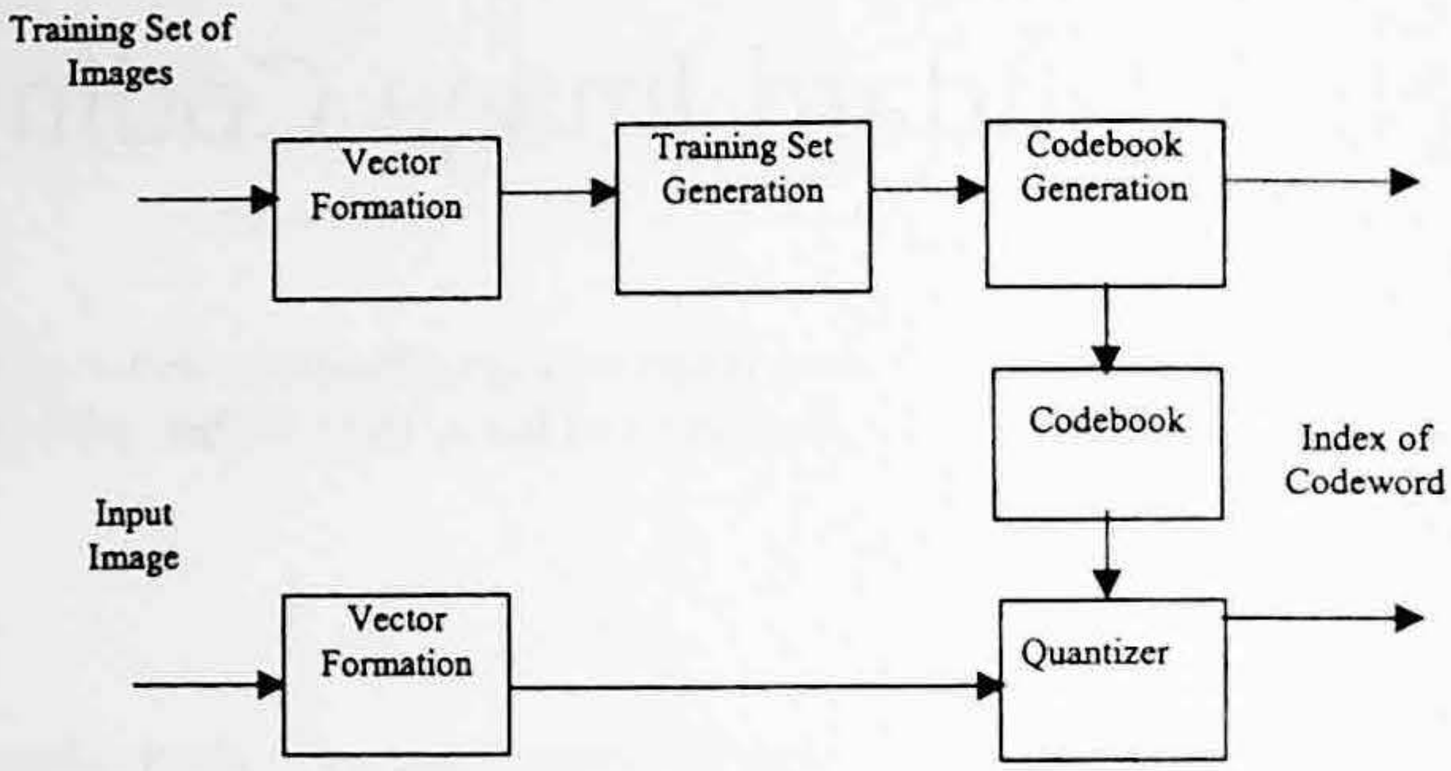


FIGURE 9.1 Principle of image vector quantization. The dashed lines correspond to training set generation, codebook generation, and transmission (if it is necessary).

model parameters are used to reconstruct the image with the same model used at the encoder. Therefore, the key techniques in the model-based coding are image modeling, image analysis, and image synthesis.

9.2 VECTOR QUANTIZATION

9.2.1 Basic Principle of Vector Quantization

An N-level vector quantizer, Q, is mapping from a K-dimensional vector set $\{V\}$, into a finite codebook, $W = \{w_1, w_2, \ldots, w_N\}$:

$$Q: V \rightarrow W \tag{9.1}$$

In other words, it assigns an input vector, v, to a representative vector (codeword), w from a codebook, W. The vector quantizer, Q, is completely described by the codebook, $W = \{w_1, w_2, \ldots, w_N\}$, together with the disjoint partition, $R = \{r_1, r_2, \ldots, r_N\}$, where

$$r_i = \{v: Q(v) = w_i\} \tag{9.2}$$

and w and v are K-dimensional vectors. The partition should identically minimize the quantization error (Gersho, 1982). A block diagram of the various steps involved in image vector quantization is depicted in Figure 9.1.

The first step in image vector quantization is the image formation. The image data are first partitioned into a set of vectors. A large number of vectors from various images are then used to form a training set. The training set is used to generate a codebook, normally using an iterative clustering algorithm. The quantization or coding step involves searching each input vector for the closest codeword in the codebook. Then the corresponding index of the selected codeword is coded and transmitted to the decoder. At the decoder, the index is decoded and converted to the corresponding vector with the same codebook as at the encoder by look-up table. Thus, the design decisions in implementing image vector quantization include (1) vector formation; (2) training set generation; (3) codebook generation; and (4) quantization.

9.2.1.1 Vector Formation

The first step of vector quantization is vector formation; that is, the decomposition of the images into a set of vectors. Many different decompositions have been proposed; examples include the

intensity values of a spatially contiguous block of pixels (Gersho and Ramamuthi, 1982; Baker and Gray, 1983); these same intensity values, but now normalized by the mean and variance of the block (Murakami et al., 1982); the transformed coefficients of the block pixels (Li and Zhang, 1995); and the adaptive linear predictive coding coefficients for a block of pixels (Sun, 1984). Basically, the approaches of vector formation can be classified into two categories: direct spatial or temporal, and feature extraction. Direct spatial or temporal is a simple approach to forming vectors from the intensity values of a spatial or temporal contiguous block of pixels in an image or an image sequence. A number of image vector quantizaton schemes have been investigated with this method. The other method is feature extraction. An image feature is a distinguishing primitive characteristic. Some features are natural in the sense that they are defined by the visual appearance of an image, while the other so-called artificial features result from specific manipulations or measurements of images or image sequences. In vector formation, it is well known that the image data in a spatial domain can be converted to a different domain so that subsequent quantization and joint entropy encoding can be more efficient. For this purpose, some features of image data, such as transformed coefficients and block means can be extracted and vector quantized. The practical significance of feature extraction is that it can result in the reduction of vector size, consequently reducing the complexity of coding procedure.

9.2.1.2 Training Set Generation

An optimal vector quantizer should ideally match the statistics of the input vector source. However, if the statistics of an input vector source are unknown, a training set representative of the expected input vector source can be used to design the vector quantizer. If the expected vector source has a large variance, then a large training set is needed. To alleviate the implementation complexity caused by a large training set, the input vector source can be divided into subsets. For example, in (Gersho, 1982) the single input source is divided into "edge" and "shade" vectors, and then the separate training sets are used to generate the separate codebooks. Those separate codebooks are then concatenated into a final codebook. In other methods, small local input sources corresponding to portions of the image are used as the training sets, thus the codebook can better match the local statistics. However, the codebook needs to be updated to track the changes in local statistics of the input sources. This may increase the complexity and reduce the coding efficiency. Practically, in most coding systems a set of typical images is selected as the training set and used to generate the codebook. The coding performance can then be insured for the images with the training set, or for those not in the training set but with statistics similar to those in the training set.

9.2.1.3 Codebook Generation

The key step in conventional image vector quantization is the development of a good codebook. The optimal codebook, using the mean squared error (MSE) criterion, must satisfy two necessary conditions (Gersho, 1982). First, the input vector source is partitioned into a predecided number of regions with the minimum distance rule. The number of regions is decided by the requirement of the bit rate, or compression ratio and coding performance. Second, the codeword or the representative vector of this region is the mean value, or the statistical center, of the vectors within the region. Under these two conditions, a generalized Lloyd clustering algorithm proposed by Linde, Buzo, and Gray (1980) — the so-called LBG algorithm — has been extensively used to generate the codebook. The clustering algorithm is an iterative process, minimizing a performance index calculated from the distances between the sample vectors and their cluster centers. The LBG clustering algorithm can only generate a codebook with a local optimum, which depends on the initial cluster seeds. Two basic procedures have been used to obtain the initial codebook or cluster seeds. In the first approach, the starting point involves finding a small codebook with only two codewords, and then recursively splitting the codebook until the required number of codewords is

obtained. This approach is referred to as binary splitting. The second procedure starts with initial seeds for the required number of codewords, these seeds being generated by preprocessing the training sets. To address the problem of a local optimum, Equitz (1989) proposed a new clustering algorithm, the pairwise nearest neighbor (PNN) algorithm. The PNN algorithm begins with a separate cluster for each vector in the training set and merges together two clusters at a time until the desired codebook size is obtained. At the beginning of the clustering process, each cluster contains only one vector. In the following process the two closest vectors in the training set are merged to their statistical mean value, in such a way the error incurred by replacing these two vectors with a single codeword is minimized. The PNN algorithm significantly reduces computational complexity without sacrificing performance. This algorithm can also be used as an initial codebook generator for the LBG algorithm.

9.2.1.4 Quantization

Quantization in the context of a vector quantization involves selecting a codeword in the codebook for each input vector. The optimal quantization, in turn, implies that for each input vector, v, the closest codeword, w_i, is found as shown in Figure 9.2. The measurement criterion could be mean squared error, absolute error, or other distortion measures.

A full-search quantization is an exhaustive search process over the entire codebook for finding the closest codeword, as shown in Figure 9.3(a). It is optimal for the given codebook, but the computation is more expensive. An alternative approach is a tree-search quantization, where the search is carried out based on a hierarchical partition. A binary tree search is shown in Figure 9.3(b). A tree search is much faster than a full search, but it is clear that the tree search is suboptimal for the given codebook and requires more memory for the codebook.

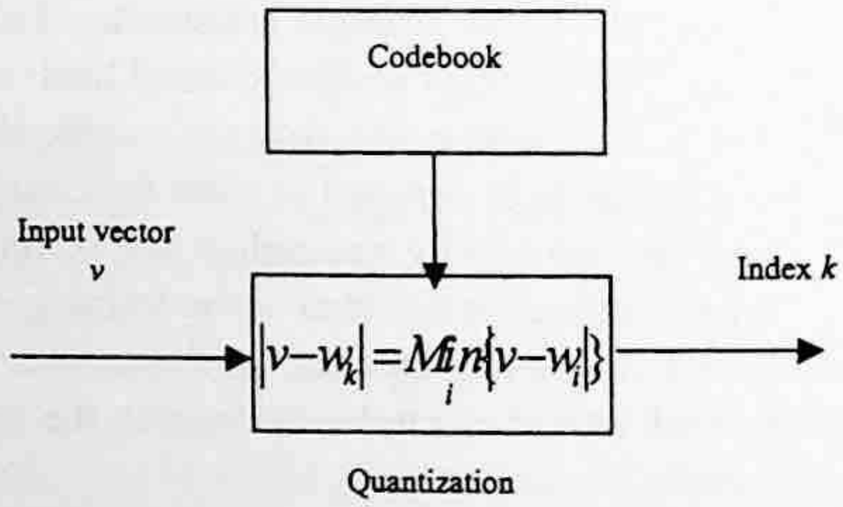


FIGURE 9.2 Principle of vector quantization.

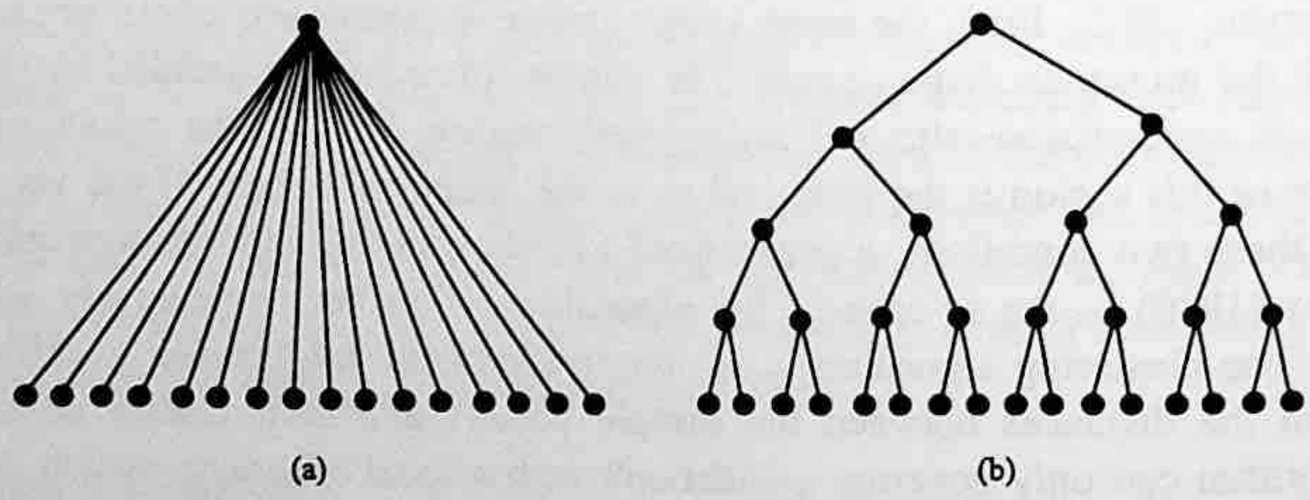


FIGURE 9.3 (a) Full search quantization; (b) binary tree search quantization.

9.2.2 Several Image Coding Schemes with Vector Quantization

In this section, we are going to present several image coding schemes using vector quantization which include residual vector quantization, classified vector quantization, transform domain vector quantization, predictive vector quantization, and block truncation coding (BTC) which can be seen as a binary vector quantization.

9.2.2.1 Residual VQ

In the conventional image vector quantization, the vectors are formed by spatially partitioning the image data into blocks of 8×8 or 4×4 pixels. In the original spatial domain the statistics of vectors may be widely spread in the multidimensional vector space. This causes difficulty in generating the codebook with a finite size and limits the coding performance. Residual VQ is proposed to alleviate this problem. In residual VQ, the mean of the block is extracted and coded separately. The vectors are formed by subtracting the block mean from the original pixel values. This scheme can be further modified by considering the variance of the blocks. The original blocks are converted to the vectors with zero mean and unit standard deviation with the following conversion formula (Murakami et al., 1982):

$$m_i = \frac{1}{K}\sum_{j=0}^{k-1} s_j \tag{9.3}$$

$$x_j = \frac{\left(s_j - m_i\right)}{\sigma_i} \tag{9.4}$$

$$\sigma_i = \left[\frac{1}{K}\sum_{j=0}^{K-1}\left(s_j - m_i\right)^2\right]^{\frac{1}{2}} \tag{9.5}$$

where m_i is the mean value of ith block, σ_i is the variance of ith block, s_j is the pixel value of pixel j ($j = 0, \ldots, K\text{-}1$) in the ith block, K is the total number of pixels in the block, and x_j is the normalized value of pixel j. The new vector X_i is now formed by x_j ($j = 0, 1, \ldots, k\text{-}1$):

$$X_i = [x_0, x_1, \ldots, x_K]_i \tag{9.6}$$

With the above normalization the probability function $P(X)$ of input vector X is approximately similar for image data from different scenes. Therefore, it is easy to generate a codebook for the new vector set. The problem with this method is that the mean and variance values of blocks have to be coded separately. This increases the overhead and limits the coding efficiency. Several methods have been proposed to improve the coding efficiency. One of these methods is to use predictive coding to code the block mean values. The mean value of the current block can be predicted by one of the previously coded neighbors. In such a way, the coding efficiency increases as the use of interblock correlation.

9.2.2.2 Classified VQ

In image vector quantization, the codebook is usually generated using training set under constraint of minimizing the mean squared error. This implies that the codeword is the statistical mean of the

region. During quantization, each input vector is replaced by its closest codeword. Therefore, the coded images usually suffer from edge distortion at very low bit rates, since edges are smoothed by the operation of averaging with the small-sized codebook. To overcome this problem, we can classify the training vector set into edge vectors and shade vectors (Gersho, 1982). Two separate codebooks can then be generated with the two types of training sets. Each input vector can be coded by the appropriate codeword in the codebook. However, the edge vectors can be further classified into many types according to their location and angular orientation. The classified VQ can be extended into a system which contains many sub-codebooks, each representing a type of edge. However, this would increase the complexity of the system and would be hard to implement in practical applications.

9.2.2.3 Transform Domain VQ

Vector quantization can be performed in the transform domain. A spatial block of 4×4 or 8×8 pixels is first transformed to the 4×4 or 8×8 transformed coefficients. There are several ways to form vectors with transformed coefficients. In the first method, a number of high-order coefficients can be discarded since most of the energy is usually contained in the low-order coefficients for most blocks. This reduces the VQ computational complexity at the expense of a small increase in distortion. However, for some active blocks, the edge information is contained in the high frequencies, or high-order coefficients. Serious subjective distortion will be caused by discarding high frequencies. In the second method, the transformed coefficients are divided into several bands and each band is used to form its corresponding vector set. This method is equivalent to the classified VQ in spatial domain. An adaptive scheme is then developed by using two kinds of vector formation methods. The first method is used for the blocks containing the moderate intensity variation and the second method is used for the blocks with high spatial activities. However, the complexity increases as more codebooks are needed in this kind of adaptive coding system.

9.2.2.4 Predictive VQ

The vectors are usually formed by the spatially consecutive blocks. The consecutive vectors are then highly statistically dependent. Therefore, better coding performance can be achieved if the correlation between vectors is exploited. Several predictive VQ schemes have been proposed to address this problem. One kind of predictive VQ is finite state VQ (Foster et al., 1985). The finite-state VQ is similar to a trellis coder. In the finite state VQ, the codebook consists of a set of sub-codebooks. A state variable is then used to specify which sub-codebook should be selected for coding the input vector. The information about the state variable must be inferred from the received sequence of state symbols and initial state such as in a trellis coder. Therefore, no side information or no overhead need be transmitted to the decoder. The new encoder state is a function of the previous encoder state and the selected sub-codebook. This permits the decoder to track the encoder state if the initial condition is known. The finite-state VQ needs additional memory to store the previous state, but it takes advantage of correlation between successive input vectors by choosing the appropriate codebook for the given past history. It should be noted that the minimum distortion selection rule of conventional VQ is not necessary optimum for finite-state VQ for a given decoder since a low-distortion codeword may lead to a bad state and hence to poor long-term behavior. Therefore, the key design issue of finite-state VQ is to find a good next-state function.

Another predictive VQ was proposed by Hang and Woods (1985). In this system, the input vector is formed in such a way that the current pixel is as the first element of the vector and the previous inputs as the remaining elements in the vector. The system is like a mapping or a recursive filter which is used to predict the next pixel. The mapping is implemented by a vector quantizer look-up table and provides the predictive errors.

9.2.2.5 Block Truncation Coding

In the block truncation code (BTC) (Delp and Mitchell, 1979), an image is first divided into 4×4 blocks. Each block is then coded individually. The pixels in each block are first converted into two-level signals by using the first two moments of the block:

$$a = m + \sigma\sqrt{\frac{q}{N-q}} \qquad (9.7)$$

$$b = m - \sigma\sqrt{\frac{N-1}{q}}$$

where m is the mean value of the block, σ is the standard deviation of the block, N is the number of total pixels in the block, and q is the number of pixels which are greater in value than m. Therefore, each block can be described by the values of block mean, variance, and a binary-bit plane which indicates whether the pixels have values above or below the block mean. The binary-bit plane can be seen as a binary vector quantizer. If the mean and variance of the block are quantized to 8 bits, then 2 bits per pixel is achieved for blocks of 4×4 pixels. The conventional BTC scheme can be modified to increase the coding efficiency. For example, the block mean can be coded by a DPCM coder which exploits the interblock correlation. The bit plane can be coded with an entropy coder on the patterns (Udpikar and Raina, 1987).

9.2.3 Lattice VQ for Image Coding

In conventional image vector quantization schemes, there are several issues, which cause some difficulties for the practical application of image vector quantization. The first problem is the limitation of vector dimension. It has been indicated that the coding performance of vector quantization increases as the vector dimension while the coding complexity exponentially increases at the same time as the increasing vector dimension. Therefore, in practice only a small vector dimension is possible under the complexity constraint. Another important issue in VQ is the need for a codebook. Much research effort has gone into finding how to generate a codebook. However, in practical applications there is another problem of how to scale the codebook for various rate-distortion requirements. The codebook generated by LBG-like algorithms with a training set is usually only suitable for a specified bit rate and does not have the flexibility of codebook scalability. For example, a codebook generated for an image with small resolution may not be suitable for images with high resolution. Even for the same spatial resolution, different bit rates would require different codebooks. Additionally, the VQ needs a table to specify the codebook and, consequently, the complexity of storing and searching is too high to have a very large table. This further limits the coding performance of image VQ.

These problems become major obstacles for implementing image VQ. Recently, an algorithm of lattice VQ has been proposed to address these problems (Li et al., 1997). Lattice VQ does not have the above problems. The codebook for lattice VQ is simply a collection of lattice points uniformly distributed over the vector space. Scalability can be achieved by scaling the cell size associated with every lattice point just like in the scalar quantizer by scaling the quantization step. The basic concept of the lattice can be found in (Conway and Slone, 1991). A typical lattice VQ scheme is shown in Figure 9.4. There are two steps involved in the image lattice VQ. The first step is to find the closest lattice point for the input vector. The second step is to label the lattice point, i.e., mapping a lattice point to an index. Since lattice VQ does need a codebook, the index assignment is based on a lattice labeling algorithm instead of a look-up table such as in conventional VQ. Therefore, the key issue of lattice VQ is to develop an efficient lattice-labeling algorithm. With this

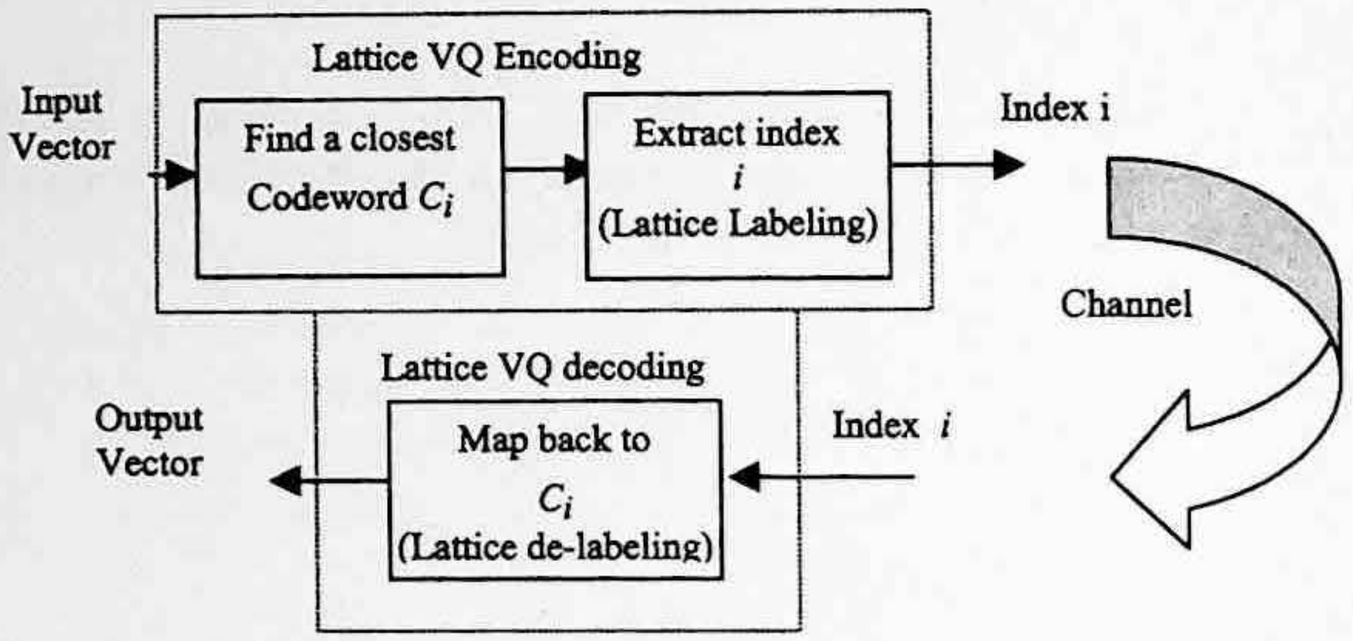


FIGURE 9.4 Block diagram of lattice VQ.

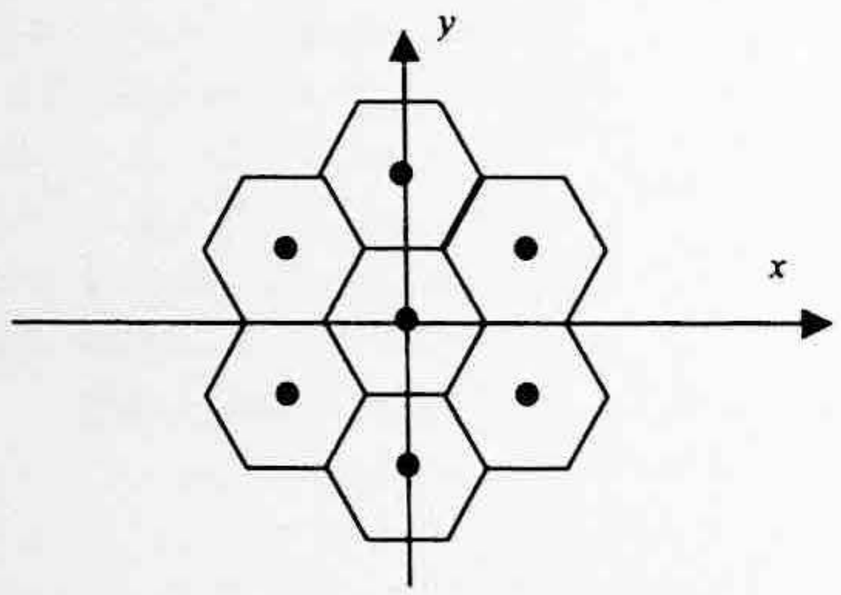


FIGURE 9.5 Labeling a two-dimensional lattice.

algorithm the closest lattice point and its corresponding index within a finite boundary can be obtained by a calculation at the encoder for each input vector.

At the decoder, the index is converted to the lattice point by the same labeling algorithm. The vector is then reconstructed with the lattice point. The efficiency of a labeling algorithm for lattice VQ is measured by how many bits are needed to represent the indices of the lattice points within a finite boundary. We use a two-dimensional lattice to explain the lattice labeling efficiency. A two-dimensional lattice is shown in Figure 9.5.

In Figure 9.5, there are seven lattice points. One method used to label these seven 2-D lattice points is to use their coordinates (x,y) to label each point. If we label x and y separately, we need two bits to label three values of x and three bits to label a possible five values of y, and need a total of five bits. It is clear that three bits are sufficient to label seven lattice points. Therefore, different labeling algorithms may have different labeling efficiency. Several algorithms have been developed for multidimensional lattice labeling. In (Conway, 1983), the labeling method assigns an index to every lattice point within a Voronoi boundary where the shape of the boundary is the same as the shape of Voronoi cells. Apparently, for different dimension, the boundaries have different shapes. In the algorithm proposed in (Laroia, 1993), the same method is used to assign an index to each lattice point. Since the boundaries are defined by the labeling algorithm, this algorithm might not achieve a 100% labeling efficiency for a prespecified boundary such as a pyramid boundary. The algorithm proposed by Fischer (1986) can assign an index to every lattice point within a prespecified pyramid boundary and achieves a 100% labeling efficiency, but this algorithm can only be used for the Z^n lattice. In a recently proposed algorithm (Wang et al., 1998), the technical breakthrough was obtained. In this algorithm a labeling method was developed for Construction-A and Construction-B lattices (Conway, 1983), which is very useful for VQ with proper vector dimensions, such as 16, and achieves 100% efficiency. Additionally, these algorithms are used for labeling lattice

points with 16 dimensions and provide minimum distortion. These algorithms were developed based on the relationship between lattices and linear block codes. Construction-A and Construction-B are the two simplest ways to construct a lattice from a binary linear block code C = (n, k, d), where n, k, and d are the length, the dimension, and the minimum distance of the code, respectively.

A Construction-A lattice is defined as:

$$\Lambda_n = C + 2Z^n \tag{9.8}$$

where Z^n is the n-dimensional cubic lattice and C is a binary linear block code. There are two steps involved for labeling a Construction-A lattice. The first is to order the lattice points according to the binary linear block code C, and then to order the lattice points associated with a particular nonzero binary codeword. For the lattice points associated with a nonzero binary codeword, two sub-lattices are considered separately. One sub-lattice consists of all the dimensions that have a "0" component in the binary codeword and the other consists of all the dimensions that have a "1" component in the binary codeword. The first sub-lattice is considered as a 2Z lattice while the second is considered as a translated 2Z lattice. Therefore, the labeling problem is reduced to labeling the Z lattice at the final stage.

A Construction-B lattice is defined as:

$$\Lambda_n = C + 2D_n \tag{9.9}$$

where D_n is an n-dimensional Construction-A lattice with the definition as:

$$D_n = (n, n-1, 2) + 2Z^n \tag{9.10}$$

and C is a binary doubly even linear block code. When n is equal to 16, the binary even linear block code associated with Λ_{16} is C = (16, 5, 8). The method for labeling a Construction-B lattice is similar to the method for labeling a Construction-A lattice with two minor differences. The first difference is that for any vector $y = c + 2x$, $x \in Z^n$, if y is a Construction-A lattice point; and $x \in D_n$, if y is a Construction-B lattice point. The second difference is that C is a binary doubly even linear block code for Construction-B lattices while it is not necessarily doubly even for Construction-A lattices. In the implementation of these lattice point labeling algorithms, the encoding and decoding functions for lattice VQ have been developed in (Li et al., 1997). For a given input vector, an index representing the closest lattice point will be found by the encoding function, and for an input index the reconstructed vector will be generated by the decoding function. In summary, the idea of lattice VQ for image coding is an important achievement in eliminating the need for a codebook for image VQ. The development of efficient algorithms for lattice point labeling makes lattice VQ feasible for image coding.

9.3 FRACTAL IMAGE CODING

9.3.1 Mathematical Foundation

A fractal is a geometric form whose irregular details can be represented by some objects with different scale and angle, which can be described by a set of transformations such as affine transformations. Additionally, the objects used to represent the image's irregular details have some form of self-similarity and these objects can be used to represent an image in a simple recursive way. An example of fractals is the Von Koch curve as shown in Figure 9.6. The fractals can be used to generate an image. The fractal image coding that is based on iterated function systems

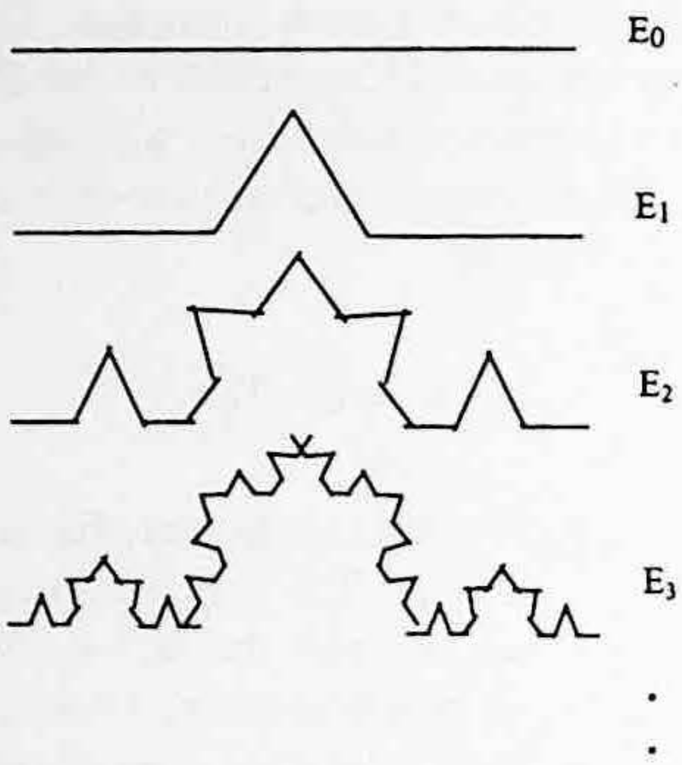


FIGURE 9.6 Construction of the Von Koch curve.

(*IFS*) is the inverse process of image generation with fractals. Therefore, the key technology of fractal image coding is the generation of fractals with an *IFS*.

To explain *IFS*, we start from the contractive affine transformation. A two-dimensional affine transformation A is defined as follows:

$$A\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix} \tag{9.11}$$

This is a transformation which consists of a linear transformation followed by a shift or translation, and maps points in the Euclidean plane into new points in the another Euclidean plane. We define that a transformation is contractive if the distance between two points P_1 and P_2 in the new plane is smaller than their distance in the original plane, i.e.,

$$d\big(A(P_1), A(P_2)\big) < s\, d(P_1, P_2) \tag{9.12}$$

where s is a constant and $0 < s < 1$. The contractive transformations have the property that when the contractive transformations are repeatedly applied to the points in a plane, these points will converge to a fixed point. *An iterated function system (IFS) is defined as a collection of contractive affine transformations*. A well-known example of *IFS* contains four following transformations:

$$A_i\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix} \quad i = 1,2,3,4. \tag{9.13}$$

This is the *IFS* of a fern leaf, whose parameters are shown in Table 9.1.

The transformation A_1 is used to generate the stalk, the transformation A_2 is used to generate the right leaf, the transformation A_3 is used to generate the left leaf, and the transformation A_4 is used to generate main fern. A fundamental theorem of fractal geometry is that each *IFS* defines a unique fractal image. This image is referred to as the attractor of the *IFS*. In other words, an image corresponds to the attractor of an *IFS*. Now let us explain how to generate the image using the *IFS*. Let us suppose that an *IFS* contains N affine transformations, $A_1, A_2, \ldots A_N$, and each transformation has an associated probability, $p_1, p_2, \ldots, p_N$, respectively. Suppose that this is a complete set and the sum of the probability equals to 1, i.e.,

TABLE 9.1
The Parameters of the *IFS* of a Fern Leaf

	a	b	c	d	e	f
A_1	0	0	0	0.16	0	0.2
A_2	0.2	–0.26	0.23	0.22	0	0.2
A_3	–0.15	0.28	0.26	0.24	0	0.2
A_4	0.85	0.04	–0.04	0.85	0	0.2

$$p_1 + p_2 + \ldots + p_N = 1 \text{ and } p_i > 0 \text{ for } i = 0, 1, \ldots, N. \tag{9.14}$$

The procedure for generating an attractor is as follows. For any given point (x_0, y_0) in a Euclidean plane, one transformation in the *IFS* according to its probability is selected and applied to this point to generate a new point (x_1, y_1). Then another transformation is selected according to its probability and applied to the point (x_1,y_1) to obtain a new point (x_2,y_2). This process is repeated over and over again to obtain a long sequence of points: (x_0,y_0), (x_1,y_1), …, (x_n,y_n), …. According to the theory of iterated function systems, these points will converge to an image that is the attractor of the given *IFS*. The above-described procedure is shown in the flowchart of Figure 9.7. With the above algorithm and the parameters in Table 9.1, initially the point can be anywhere within the large square, but after several iterations it will converge onto the fern. The 2-D affine transformations are extended to 3-D transformations, which can be used to create fractal surfaces with the iterated function systems. This fractal surface can be considered as the gray level or brightness of a 2-D image.

9.3.2 *IFS*-Based Fractal Image Coding

As described in the last section, an *IFS* can be used to generate a unique image, which is referred to as an attractor of the *IFS*. In other words, this image can be simply represented by the parameters of the *IFS*. Therefore, if we can use an inverse procedure to generate a set of transformations, i.e.,

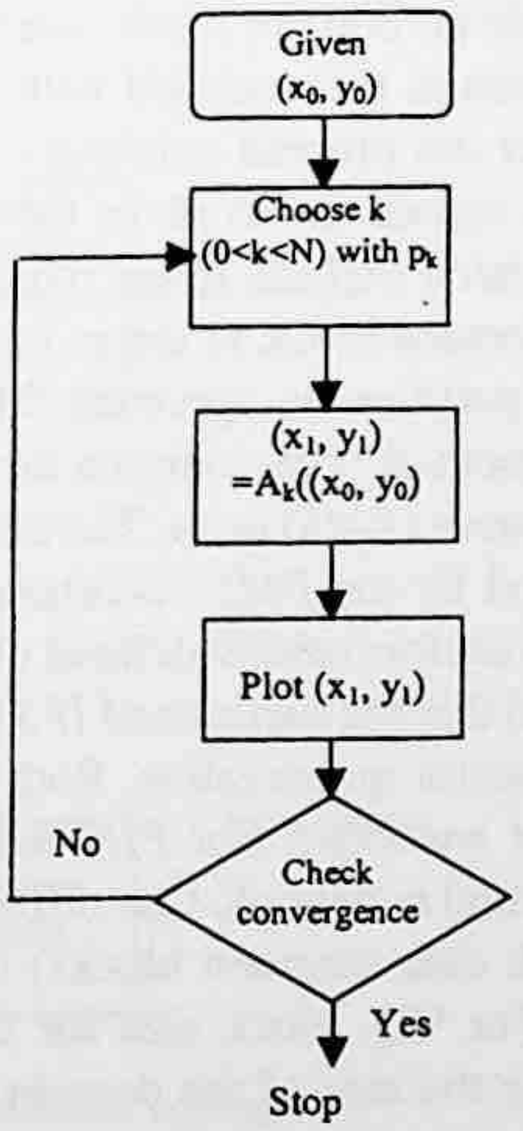


FIGURE 9.7 Flowchart of image generation with an *IFS*.

an *IFS* from an image, then these transformations or the *IFS* can be used to represent the approximation of the image. The image coding system can use the parameters of the transformations in the *IFS* instead of the original image data for storage or transmission. Since the *IFS* contains only very limited data such as transformation parameters, this image coding method may result in a very high compression ratio. For example, the fern image is represented by 24 integers or 192 bits (if each integer is represented by 8 bits). This number is much smaller than the number needed to represent the fern image pixel by pixel. Now the key issue of the *IFS*-based fractal image coding is to generate the *IFS* for the given input image. Three methods have been proposed to obtain the *IFS* (Lu, 1993). One is the direct method, that directly finds a set of contractive affine transformations from the image based on the self-similarity of the image. The second method is to partition an image into the smaller objects whose *IFS*s are known. These *IFS*s are used to form a library. The encoding procedure is to look for an *IFS* from the library for each small object. The third method is called partitioned *IFS* (*PIFS*). In this method, the image is first divided into smaller blocks and then the *IFS* for each block is found by mapping a larger block into a small block.

In the direct approach, the image is first partitioned into nonoverlapped blocks in such a way that each block is similar to the whole image and a transformation can map the whole image to the block. The transformation for each individual block may be different. The combination of these transformations can be taken as the *IFS* of the given image. Then much fewer data are required to represent the *IFS* or the transformations than to transmit or store the given image in the pixel by pixel way. For the second approach, the key issue is how to partition the given image into objects whose *IFS*s are known. The image processing techniques such as color separation, edge detection, spectrum analysis, and texture variation analysis can be used for image partitioning. However, for natural images or arbitrary images, it may be impossible or very difficult to find an *IFS* whose attractor perfectly covers the original image. Therefore, for most natural images the partitioned *IFS* method has been proposed (Lu, 1993). In this method, the transformations do not map the whole image into small block. For encoding an image, the whole image is first partitioned into a number of larger blocks that are referred to as domain blocks. The domain blocks can be overlapped. Then the image is partitioned into a number of smaller blocks that are called as range blocks. The range blocks do not overlap and the sum total of the range blocks covers the whole image. In the third step, a set of contractive transformations is chosen. Each range block is mapped into a domain block with a searching method and a matching criterion. The combination of the transformations is used to form a partitioned *IFS* (*PIFS*). The parameters of *PIFS* are transmitted to the decoder. It is noted that no domain blocks are transmitted. The decoding starts with a flat background. The iterated process is then applied with the set of transformations. The reconstructed image is then obtained after the process converges. From the above discussion, it is found that there are three main design issues involved in the block fractal image coding system. First are partitioning techniques which include range block partitioning and domain block partitioning. As mentioned earlier, the domain block is larger than the range block. Dividing the image into square blocks is the simplest partitioning approach. The second issue is the choice of distortion measurement and a searching method. The common distortion measurement in the block fractal image coding is the root mean square (*RMS*) error. The closest match between the range block and transformed domain block is found by the *RMS* distortion measurement. The third method is the selection of a set of contractive transformations defined consistently with a partition.

It is noted that the partitioned *IFS* (*PIFS*)-based fractal image coding has several similar features with image vector quantization. Both coding schemes are block-based coding schemes and need a codebook for encoding. For *PIFS*-based fractal image coding the domain blocks can be seen as forming a virtual codebook. One difference is that the fractal image coding does not need to transmit the codebook data (domain blocks) to the decoder while VQ does. The second difference is the block size. For VQ, block size for the code vector and input vector is the same while in *PIFS* fractal coding the size of the domain block is different from the size of the range blocks. Another

difference is that in fractal image coding the image itself serves as the codebook, while this is not true for VQ image coding.

9.3.3 Other Fractal Image Coding Methods

Besides the *IFS*-based fractal image coding, there are several other fractal image coding methods. One is the segmentation-based coding scheme using fractal dimensions. In this method, the image is segmented into regions based on the properties of the human visual system (*HVS*). The image is segmented into the regions, each of these regions is homogeneous in the sense of having similar features by visual perception. This is different from the traditional image segmentation techniques that try to segment an image into regions of constant intensity. For a complicated image, good representation of an image needs a large number of small segmentations. However, in order to obtain a high compression ratio, the number of segmentations is limited. The trade-off between image quality and bit rate has to be considered. A parameter, fractal dimension, is used as a measure to control the trade-off. Fractal dimension is a characteristic of a fractal. It is related to a metric property such as the length of a curve and the area of a surface. The fractal dimension can provide a good measurement of the perceptual roughness of the curve and surface. For example, if we use many segments of straight lines to approximate a curve, by increasing the length of the straight lines perceptually rougher curves are represented.

9.4 MODEL-BASED CODING

9.4.1 Basic Concept

In the model-based coding, an image model that can be a 2-D model for still images or a 3-D model for video sequence is first constructed. At the encoder, the model is used to analyze the input image. The model parameters are then transmitted to the decoder. At the decoder the reconstructed image is synthesized by the model parameters, with the same image model used at the encoder. This basic idea of model-based coding is shown in the Figure 9.8. Therefore, the basic techniques in the model-based coding are the image modeling, image analysis, and image synthesis techniques. Both image analysis and synthesis are based on the image model. The image modeling techniques used for image coding can normally be divided into two classes: structure modeling and motion modeling. Motion modeling is usually used for video sequences and moving pictures, while structure modeling is usually used for still image coding. The structure model is used for reconstruction of a 2-D or 3-D scene model.

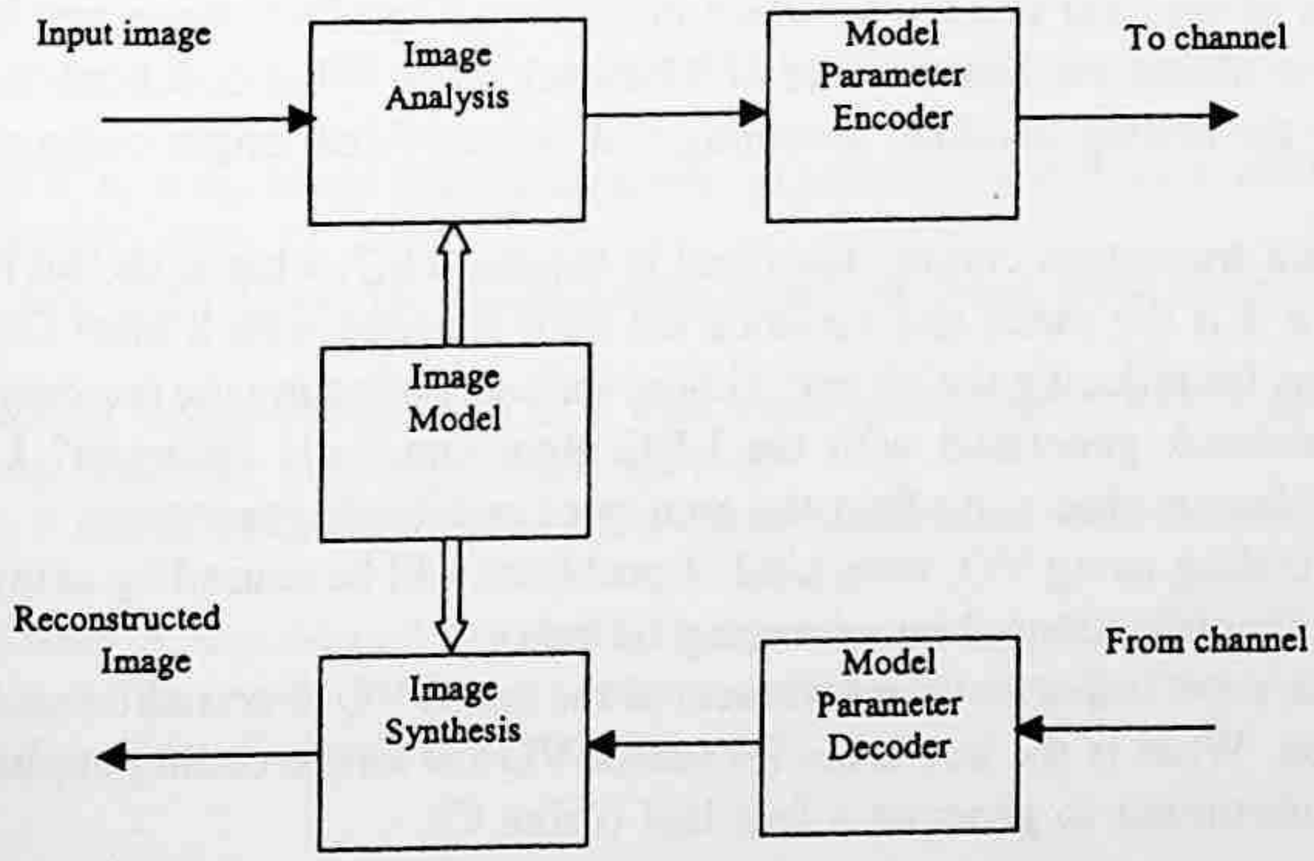


FIGURE 9.8 Basic principle of model-based coding.

9.4.2 Image Modeling

The geometric model is usually used for image structure description. The geometric model can be classified into a surface-based description and volume-based description. The major advantage of surface description is that such description is easily converted into a surface representation that can be encoded and transmitted. In these models the surface is approximated by planar polygonal patches such as triangle patches. The surface shape is represented by a set of points that represent the vertices of these triangle meshes. The size of these triangle patches can be adjusted according to the surface complexity. In other words, for more complicated areas, more triangle meshes are needed to approximate the surface while for smoothing areas, the mesh sizes can be larger or less vertices of the triangle meshes are needed to represent the surface. The volume-based description is a natural approach for modeling most solid world objects. Most existing research work on volume-based description focuses on the parametric volume description. The volume-based description is used for 3-D objects or video sequences.

However, model-based coding is successfully applicable only to certain kinds of images since it is very hard to find general image models suitable for most natural scenes. The few successful examples of image models include the human face, head, and body. These models are developed for the analysis and synthesis of moving images. The face animation has been adopted for the MPEG-4 visual coding. The body animation is under consideration for version 2 of MPEG-4 visual coding.

9.5 SUMMARY

In this chapter three kinds of image coding techniques, vector quantization, fractal image coding, and model-based coding, which are not used in the current standards, have been presented. All three techniques have several important features such as very high compression ratios for certain kinds of images and very simple decoding procedures (especially for VQ). However, due to some limitations these techniques have not been adopted by industry standards. It should be noted that recently the facial model face animation technique has been adopted for the MPEG-4 visual standard (mpeg4 visual).

9.6 EXERCISES

9-1. In the modified residual VQ described in Equation 9.5, with a 4×4 block and 8 bits for each pixel of original image, we use 8 bits for coding block mean and block variance. We want to obtain the final bit rate of 2 bits per pixel. What codebook size do we have to use for the coding residual, assuming that we use fixed-length coding to code vector indices?

9-2. In the block truncation coding described in Equation 9.7, what is the bit rate for a block size of 4×4 if the mean and variance are both encoded with 8 bits? Do you have any suggestions for reducing the bit rate without seriously affecting the reconstruction quality?

9-3. Is the codebook generated with the LBG algorithm local optimum? List the several important factors that will affect the quality of codebook generation.

9-4. In image coding using VQ, what kind of problems will be caused by using the codebook in practical applications (hint: changing bit rate).

9-5. What is the most important improvement of the lattice VQ over traditional VQ in practical application. What is the key issue for lattice VQ for image coding application?

9-6. Write a subroutine to generate a fern leaf (using C).

REFERENCES

Baker, R. L. and R. M. Gray, Image compression using nonadaptive spatial vector quantization, ISCAS'83, 1983, 55-61.

Barnsley, M.F. and A.E. Jacquin, Application of recurrent iterated function systems, SPIE, vol. 1001, *Visual Communications and Image Processing*, 1988, 122-131.

Barnsley, M. and L. P. Hurd, *Fractal Image Compression*, A.K. Peters, Wellesley, MA, 1993.

Conway, J. H. and N. J. A. Slone, A fast encoding method for lattice codes and quantizers, *IEEE Trans. Inform. Theory*, vol. IT-29, 820-824, 1983.

Conway, J. H. and N. J. A. Slone, *Sphere Packings, Lattices and Groups*, New York: Springer-Verlag, 1991.

Delp, E. J. and D. R. Mitchell, Image compression using block truncation coding, *IEEE Trans. Commun.*, COM-27, 1979.

Dunham, M. and R. Gray, An algorithm for the design of labelled-transition finite-state vector quantizer, *IEEE Trans. Commun.*, COM-33, 83-89, 1985.

Equits, W. H. A new vector quantization clustering algorithm, *IEEE Trans. Acoust. Speech Signal Process.*, 37, 1568-1575, 1989.

Fischer, T. R., A paramid vector quantization, *IEEE Trans. Inform. Theory*, vol. IT-32, 568-583, 1986.

Fisher, Y. *Fractal Image Compression — Theory and Application,* New York: Springer-Verlag, 1994.

Foster, J., R. M. Gray, and M. O. Dunham, Finite-state vector quantization for waveform coding, *IEEE Trans. Inf. Theory,* IT-31, 348-359, 1985.

Gersho, A. and B. Ramamurthi, Image coding using vector quantization, ICASSP'82, Paris, May 1982, 428-431.

Gersho, A. On the structure of vector quantizer, *IEEE Trans. Inf. Theory,* IT-28, 157-166, 1982.

Hang, H. M. and J. W. Woods, Predictive vector quantization of images, *IEEE Trans. Commun.*, COM-33, 1208-1219, 1985.

ISO/IEC 14496-2, Coding of Audio-Visual Objects, Part 2, Dec. 18, 1998.

Jacquin, A. E. Fractal Image Coding: A Review, *Proc. IEEE,* 81(10), 1451-1465, 1993.

Jang, J. and S. A. Rajala, Segmentation-based image coding using fractals and the human visual system, IEEE Int. Conf. Acoust. Speech Signal Processing, 1990, pp. 1957-1960.

Laroia, R. and N. Farvardin, A structured fixed rate vector quantizer derived from a variable length scaler quantizer: I & II, *IEEE Trans. Inform. Theory*, vol. 39, 851-876, 1993.

Li, H., A. Lundmark, and R. Forchheimer, Image Sequence Coding at Very Low Bitrates: A Review, *IEEE Trans. Image Process.*, 3(5), 1994.

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

Li, W. et al., A video coding algorithm using vector-based tehnique, *IEEE Trans. Circuits Syst. Video Technol.*, 7(1), 146-157, 1997.

Linde, Y. A. Buzo, and R. M. Gray, An algorithm for vector quantizer design, *IEEE Trans. Commun.*, 28, 84-95, 1980.

Lu, G. Fractal image compression, *Signal Process. Image Commun.*, 5, 327-343, 1993.

Murakami, T., K. Asai, and E. Yamazaki, Vector quantization of video signals, *Electron. Lett.*, 7, 1005-1006, 1982.

Nasrabadi, N. M. and R. A. King, Image Coding using Vector Quantization: A Review, *IEEE Trans. Commun.*, COM-36(8), 957-971, 1988.

Stewart, L. C., R. M. Gray and Y. Linde, The design of trellis waveform coders, *IEEE Trans. Commun.*, COM-30, 702-710, 1982.

Sun, H. and M. Goldberg, Image coding using LPC with vector quantization, *IEEE Proc. Int. Conf. Digital Signal Processing*, Florence, Italy, Sept. 1984, 508-512.

Udpikar, V. R. and J. P. Raina, BTC image coding using vector quantization, *IEEE Trans. Commun.*, COM-35, 352-356, 1987.

Walach, E. and E. Karnin, A fractal-based approach to image compression, *ICASSP 1986*, 529-532.

Wang, C., H. Q. Cao, W. Li, and K. K. Tzeng, Lattice Labeling Algorithm for Vector Quantization, *IEEE Trans. Circuits Syst. Video Technol.*, 8(2), 206-220, 1998.

Section III

Motion Estimation and Compression

10 Motion Analysis and Motion Compensation

Up to this point, what we have discussed in the previous chapters were basic techniques in image coding, specifically, techniques utilized in still image coding. From here on, we are going to address the issue of video sequence compression. To fulfill the task, we will first define the concepts of image and video sequences. Then we address the issue of interframe correlation between successive frames. Two techniques in exploitation of interframe correlation, frame replenishment and motion-compensated coding, will then be discussed. The rest of the chapter covers the concepts of motion analysis and motion compensation in general.

10.1 IMAGE SEQUENCES

In this section the concept of various image sequences is defined in a theoretical and systematic manner. The relationship between image sequences and video sequences is also discussed.

It is well known that in the 1960s the advent of the semiconductor computer and the space program swiftly brought the field of digital image processing into public focus. Since then the field has experienced rapid growth and has entered every aspect of modern technology. Since the early 1980s, digital image sequence processing has been an attractive research area (Huang, 1981a, 1983). This is not surprising, because an image sequence, as a collection of images, may provide more information than a single image frame. The increased computational complexity and memory space associated with image sequence processing are becoming more affordable due to more advanced, achievable computational capability. With the tremendous advancements continuously made in VLSI computer and information processing, image and video sequences are evermore indispensable elements of modern life. While the pace and the future of this development cannot be predicted, one thing is certain: this process is going to drastically change all aspects of our world in the next several decades.

As far as image sequence processing is concerned, it is noted that in addition to temporal image sequences, stereo image pair and stereo image sequences also received attention in the middle of the 1980s (Waxman and Duncan, 1986). The concepts of temporal and spatial image sequences, and the imaging space (which may be considered as a next-higher-level unification of temporal and spatial image sequences) may be illustrated as follows.

Consider a sensor located in a specific position in the three-dimensional (3-D) world space. It generates images about the scene, one after another. As time goes by, the images form a sequence. The set of these images can be represented with a brightness function g(x,y,t), where x and y are coordinates on the image planes. This is referred to as a *temporal image sequence*. This is the basic outline about the brightness function g(x,y,t) dealt with by researchers in both computer vision, e.g., Horn and Schunck (1980) and signal processing fields, e.g., Pratt (1979).

Now consider a generalization of the above basic outline. A sensor, as a solid article, can be translated (in three free dimensions) and rotated (in two free dimensions). It is noted that here the rotation of a sensor about its optical axis is not counted, since the images generated will remain unchanged when this type of rotation takes place. So, we can obtain a variety of images when a sensor is translated to different coordinates and rotated to different angles in the 3-D world space. Equivalently, we can imagine that there is an infinite number of sensors in the 3-D world space

that occupies all possible spatial coordinates and assumes all possible orientations at each coordinate; i.e., they are located on all possible positions. At one specific moment, all of these images form a set, which can be referred to as a *spatial image sequence*. When time varies, these sets of images form a much larger set of images, called an *imaging space*.

Clearly, it is impossible to describe such a set of images by using the above-mentioned g(x,y,t). Instead, it should be described by a more general brightness function,

$$g(x, y, t, \bar{s}), \tag{10.1}$$

where $\bar{s}$ indicates the sensor's position in the 3-D world space; i.e., the coordinates of the sensor center and the orientation of the optical axis of the sensor. Hence $\bar{s}$ is a 5-D vector. That is,

$$\bar{s} = (\tilde{x}, \tilde{y}, \tilde{z}, \beta, \gamma), \tag{10.2}$$

where $\tilde{x}$, $\tilde{y}$, and $\tilde{z}$ represent the coordinates of the optical center of the sensor in the 3-D world space; and β and γ represent the orientation of the optical axis of the sensor in the 3-D world space. More specifically, each sensor in the 3-D world space may be considered associated with a 3-D Cartesian coordinate system such that its optical center is located on the origin and its optical axis is aligned with the *OZ* axis. In the 3-D world space we choose a 3-D Cartesian coordinate system as the reference coordinate system. Hence, a sensor with its Cartesian coordinate system coincident with the reference coordinate system has its position in the 3-D world space denoted by $\bar{s}$ = (0,0,0,0,0). An arbitrary sensor position denoted by $\bar{s}$ = ($\tilde{x}$, $\tilde{y}$, $\tilde{z}$, â, ã) can be described as follows. The sensor's associated Cartesian coordinate system is first shifted from the reference coordinate system in the 3-D world space with its origin settled at ($\tilde{x}$, $\tilde{y}$, $\tilde{z}$) in the reference coordinate system. Then it is rotated with the rotation angles β and γ being the same as Euler angles (Shu and Shi, 1991; Shi et al., 1994). Figure 10.1 shows the reference coordinate system and an arbitrary Cartesian coordinate system (indicating an arbitrary sensor position). There, *oxy* and *o'x'y'* represent, respectively, the related image planes.

Assume now a world point *P* in the 3-D space that is projected onto the image plane as a pixel with the coordinates x_p and y_p. Then, x_p and y_p are also dependent on t and $\bar{s}$. That is, the coordinates of the pixel can be denoted by $x_p = x_p(t,\bar{s})$ and $y_p = y_p(t, \bar{s})$. So generally speaking, we have

$$g = g(x_p(t,\bar{s}), y_p(t,\bar{s}), t, \bar{s}). \tag{10.3}$$

As far as temporal image sequences are concerned, let us take a look at the framework of Pratt (1979), and Horn and Schunck (1980). There, $g = g(x_p(t), y_p(t), t)$ is actually a special case of Equation 10.3, i.e., $g = g(x_p(t, \bar{s} = \text{constant vector}), y_p(t, \bar{s} = \text{constant vector}), (t, \bar{s} = \text{constant vector})$. In other words, the variation of $\bar{s}$ is restricted to be zero, i.e., $\Delta\bar{s} = 0$. This means the sensor is fixed in a certain position in the 3-D world space.

Obviously, an alternative is to define the imaging space as a set of all temporal image sequences; i.e., those taken by sensors located at all possible positions in the 3-D world space. Stereo image sequences can thus be viewed as a proper subset of the imaging space, just like a stereo pair of images can be considered as a proper subset of a spatial image sequence.

In summary, the imaging space is a collection of all possible forms assumed by the general brightness function $g(x, y, t, \bar{s})$. Each picture taken by a sensor located on a particular position at a specific moment is merely a special cross section of this imaging space. Both temporal and spatial image sequences are special proper subsets of the imaging space. They are in the middle level, between the imaging space and the individual images. This hierarchical structure is depicted in Figure 10.2.

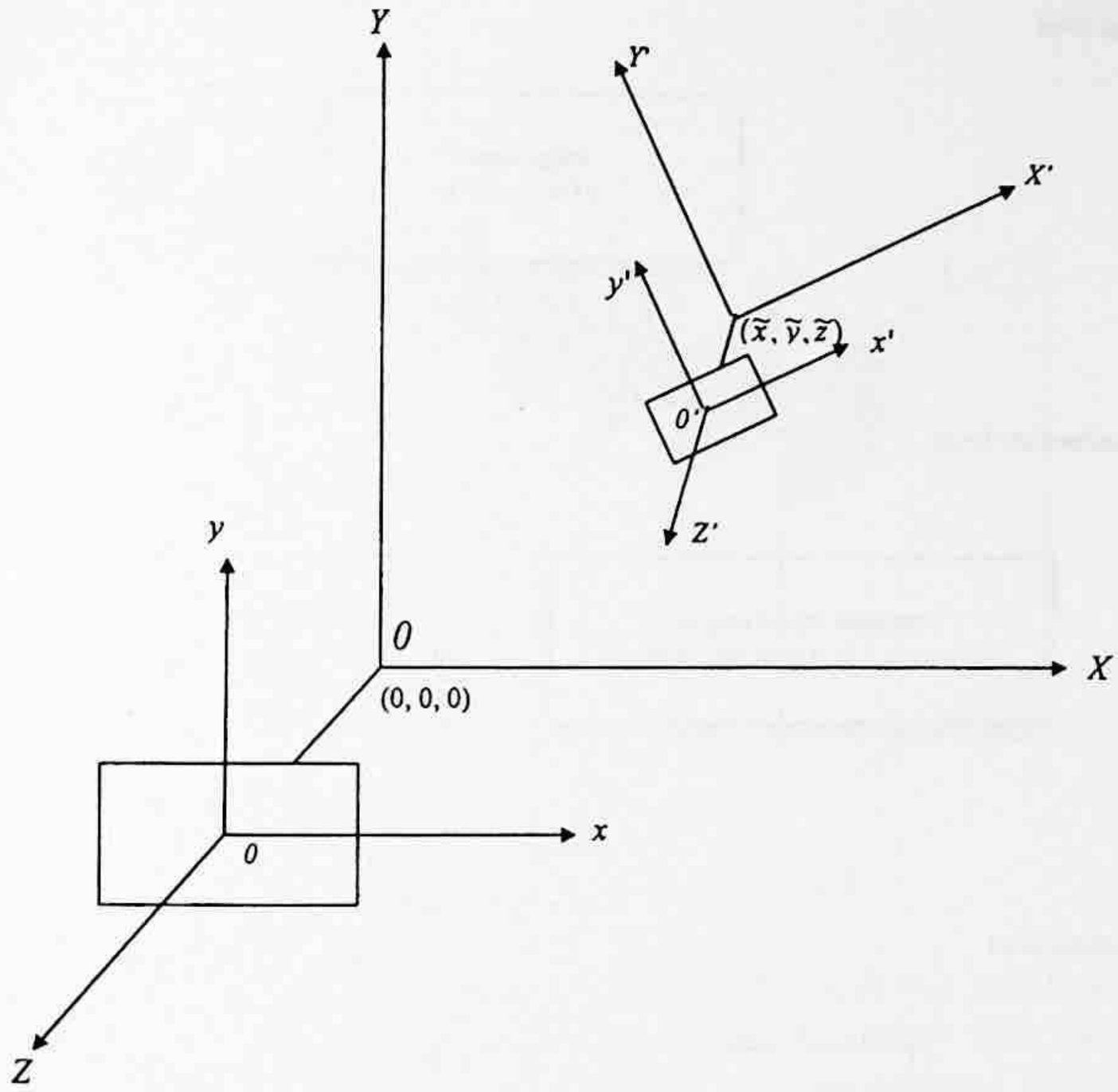


FIGURE 10.1 Two sensor positions $\bar{s} = (0,0,0,0,0)$ and $\bar{s} = (\tilde{x}, \tilde{y}, \tilde{z}, \hat{a}, \tilde{a})$.

Before we conclude this section, we should discuss the relationship between image sequences and video sequences. It is noted that the term *video* is used very often nowadays in addition to the terms *image frames* and *image sequence*. It is necessary to pause for a while to discuss the relationship between these terms. Image frames and image sequence have been defined clearly above with the introduction of the concept of the imaging space. Video can mean an individual video frame or video sequences. It refers, however, to those frames and sequences that are associated with the visible frequency band in the electromagnetic spectrum. For image frames and image sequences, there is no such restriction. For instance, infrared image frames and sequences correspond to a band outside the visible band in the spectrum. From this point of view, the scope of image frames and sequences is wider than that of video frames and sequences. When the visible band is concerned, the terms *image frame and sequence* are interchangeable with that *video frame and sequence*.

Another point we would like to bring to the reader's attention is as follows. Though video is referred to as visual information, which includes both a single frame and frame sequences, in practice it is often used to mean sequences exclusively. Such an example can be found in *Digital Video Processing* (Tekalp, 1995).

In this book, we use *image compression* to indicate still image compression, and *video compression* to indicate video sequence compression. Readers should keep in mind, however, that (1) video can mean a single frame or sequences of frames; and (2) the scope of image is wider than that of video, and video is more pertinent to multimedia engineering.

10.2 INTERFRAME CORRELATION

As far as video compression is concerned, all the techniques discussed in the previous chapters are applicable. By this we mean two classes of techniques. The first class, which is also the most straightforward way to handle video compression, is to code each frame separately. That is,

FIGURE 10.2 A hierarchical structure.

individual frames are coded independently on each other. For instance, using a JPEG compression algorithm to code each frame in a video sequence results in *motion JPEG* (Westwater and Furht, 1997). In the second class, methods utilized for still image coding can be generalized for video compression. For instance, (DCT) transform coding can be generalized and applied to video coding by extending 2-D DCT to 3-D DCT. That is, instead of 2-D DCT, say, 8×8, applied to a single image frame, we can apply 3-D DCT, say, $8 \times 8 \times 8$, to a video sequence. Refer to Figure 10.3. That is, 8 blocks of 8×8 each located, respectively, at the same position in one of the 8 successive frames from a video sequence are coded together with the 3-D DCT. It was reported that this 3-D DCT technique is quite efficient (Lim, 1990; Westwater and Furht, 1997). In addition, the DPCM technique and the hybrid technique can be generalized and applied to video compression in a similar fashion (Jain, 1989; Lim, 1990). It is noted that in the second class of techniques several successive frames are grouped and coded together, while in the first class each frame is coded independently.

Video compression has its own characteristics, however, that make it quite different from still image compression. The major difference lies in the exploitation of interframe correlation that exists between successive frames in video sequences, in addition to the intraframe correlation that exists within each frame. As mentioned in Chapter 1, the interframe correlation is also referred to as *temporal redundancy*, while the intraframe correlation is referred to as *spatial redundancy*. In order to achieve coding efficiency, we need to remove these redundancies for video compression. To do so we must first understand these redundancies.

Consider a video sequence taken in a videophone service. There, the camera is static most of the time. A typical scene is a head-and-shoulder view of a person imposed on a background. In this type of video sequence the background is usually static. Only the speaker is experiencing motion, which is not severe. Therefore, there is a strong similarity between successive frames, that

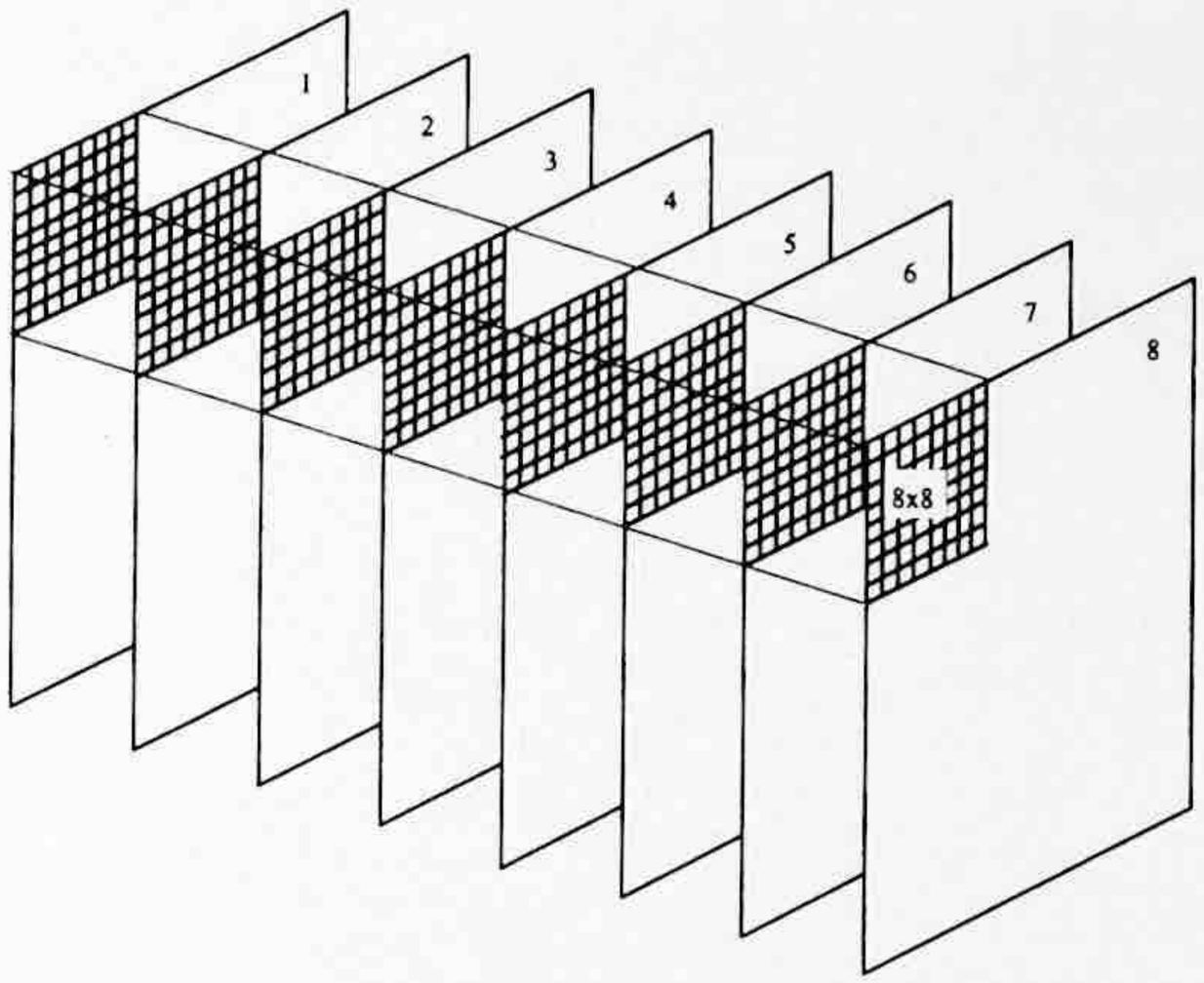


FIGURE 10.3 3-D DCT of $8 \times 8 \times 8$.

is, a strong adjacent-frame correlation. In other words, there is a strong interframe correlation. It was reported by Mounts (1969) that when using videophone-like signals with moderate motion in the scene, on average, less than one-tenth of the elements change between frames by an amount which exceeds 1% of the peak signal. Here, a 1% change is regarded as significant. Our experiment on the first 40 frames of the Miss America sequence supports this observation. Two successive frames of the sequence, frames 24 and 25, are shown in Figure 10.4.

Now, consider a video sequence generated in a television broadcast. It is well known that television signals are generated with a scene scanned in a particular manner in order to maintain a steady picture for a human being to view, regardless of whether there is a scenery change or not. That is, even if there is no change from one frame to the next, the scene is still scanned constantly. Hence there is a great deal of frame-to-frame correlation (Haskell et al., 1972b; Netravali and Robbins, 1979). In TV broadcasts, the camera is most likely not static, and it may be panned, tilted, and zoomed. Furthermore, more movement is involved in the scene. As long as the TV frames are taken

FIGURE 10.4 Two frames of the Miss America sequence: (a) frame 24, (b) frame 25.

densely enough, then most of the time we think the changes between successive frames are due mainly to the apparent motion of the objects in the scene that takes place during the frame intervals. This implies that there is also a high correlation between sequential frames. In other words, there is an interframe redundancy (interpixel redundancy between pixels in successive frames). There is more correlation between television picture elements along the frame-to-frame temporal dimension than there is between adjacent elements in a single frame along the spatial dimension. That is, there is generally more interframe correlation than intraframe correlation. Taking advantage of the interframe correlation, i.e., eliminating or decreasing the uncertainty of successive frames, leads to video data compression. This is analogous to the case of still image coding with the DPCM technique, where we can predict part of an image by knowing the other part. Now the knowledge of the previous frames can remove the uncertainty of the next frame. In both cases, knowledge of the past removes the uncertainty of the future, leaving less actual information to be transmitted (Kretzmer, 1952). In Chapter 16, we will see that the words "past" and "future" used here are not necessary. They can be changed, respectively, to "some frames" and "some other frames" in advanced video coding techniques such as MPEG. There, a frame might be predicted from both its previous frames and its future frames.

At this point, it becomes clear that the second class of techniques (mentioned at the beginning of this section), which generalizes techniques originally developed for still image coding and applies them to video coding, exploits interframe correlation. For instance, in the case of the 3-D DCT technique, a strong temporal correlation causes an energy compaction within the low temporal frequency region. The 3-D DCT technique drops transform coefficients associated with high temporal frequency, thus achieving data compression.

The two techniques specifically developed to exploit interframe redundancy, i.e., frame replenishment and motion-compensated coding, are introduced below. The former is the early work, while the latter is the more popular recent work.

10.3 FRAME REPLENISHMENT

As mentioned in Chapter 3, frame-to-frame redundancy has long been recognized in TV signal compression. The first few experiments of a frame sequence coder exploiting interframe redundancy may be traced back to the 1960s (Seyler, 1962, 1965; Mounts, 1969). In (Mounts, 1969) the first real demonstration was presented and was termed *conditional replenishment*. This frame replenishment technique can be briefly described as follows. Each pixel in a frame is classified into *changing* or *unchanging* areas depending on whether or not the intensity difference between its present value and its previous one (the intensity value at the same position on the previous frame) exceeds a threshold. If the difference does exceed the threshold, i.e., a *significant* change has been identified, the address and intensity of this pixel are coded and stored in a buffer and then transmitted to the receiver to replenish intensity. For those unchanging pixels, nothing is coded and transmitted. Their previous intensities are repeated in the receiver. It is noted that the buffer is utilized to make the information presented to the transmission channel occur at a smooth bit rate. The threshold is to make the average replenishment rate match the channel capacity.

Since the replenishment technique only encodes those pixels whose intensity value has changed significantly between successive frames, its coding efficiency is much higher than the coding techniques which encode every pixel of every frame, say, the DPCM technique applied to each single frame. In other words, utilizing interframe correlation, the replenishment technique achieves a lower bit rate, while keeping the equivalent reconstructed image quality.

Much effort had been made to further improve this type of simple replenishment algorithm. As mentioned in the discussion of 3-D DPCM in Chapter 3, for instance, it was soon realized that intensity values of pixels in a changing area need not be transmitted independently of one another. Instead, using both spatial and temporal neighbors' intensity values to predict the intensity value

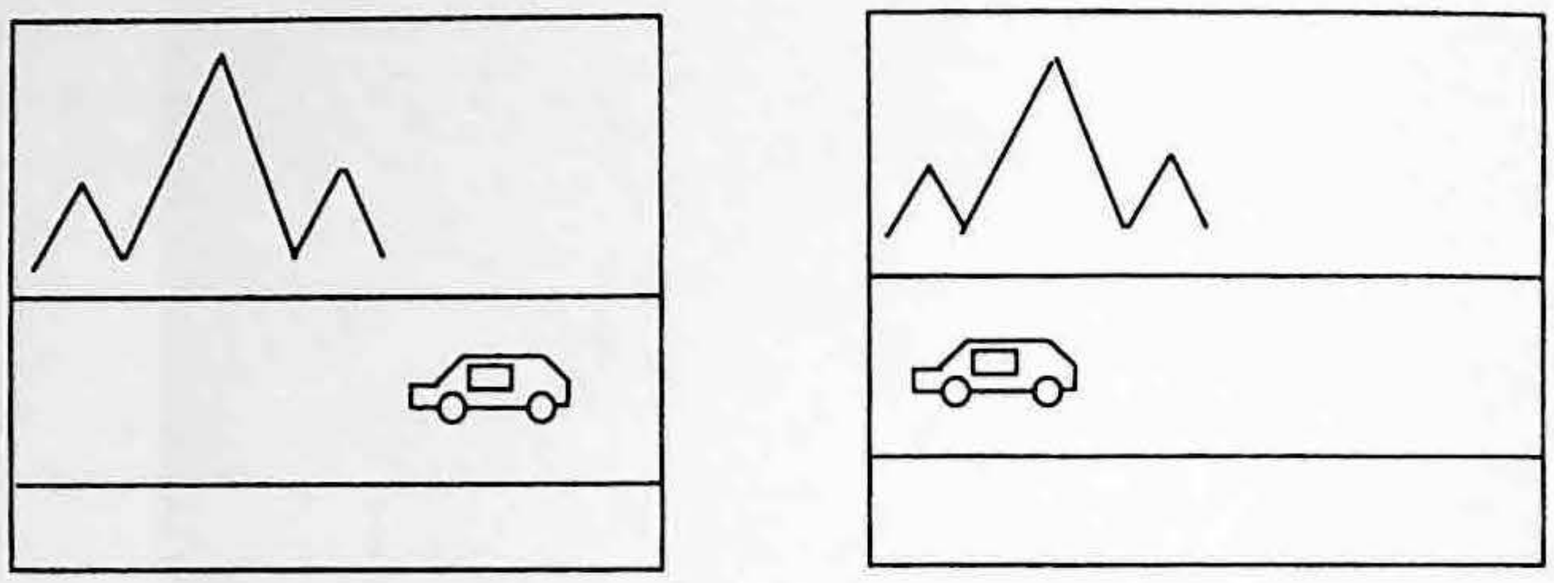
FIGURE 10.5 Dirty window effect.

of a changing pixel leads to a *frame-difference* predictive coding technique. There, the differential signal is coded instead of the original intensity values, thus achieving a lower bit rate. For more detail, readers are referred to Section 3.5.2. Another example of the improvements is that measures have been taken to distinguish the intensity difference caused by noise from those associated with changing to avoid the dirty window effect, whose meaning is given in the next paragraph. For more detailed information on these improvements over the simple frame replenishment technique, readers are referred to two excellent reviews by Haskell et al. (1972b, 1979).

The main drawback associated with the frame replenishment technique is that it is difficult to handle frame sequences containing more rapid changes. When there are more rapid changes, the number of pixels whose intensity values need to be updated increases. In order to maintain the transmission bit-rate at a steady and proper level the threshold has to be raised, thus causing many slow changes that cannot show up in the receiver. This poorer reconstruction in the receiver is somewhat analogous to viewing a scene through a dirty window. This is referred to as the dirty window effect. The result of one experiment on the dirty window effect is displayed in Figure 10.5. From frame 22 to frame 25 of the Miss America sequence, there are 2166 pixels (less than 10% of the total pixels) that change their gray level values by more than 1% of the peak signal. When we only update the gray level values for 25% (randomly chosen) of these changing pixels, we can clearly see the dirty window effect. When rapid scene changes exceed a certain level, buffer saturation will result, causing picture breakup (Mounts, 1969). Motion-compensated coding, which is discussed below, has been proved to be able to provide better performance than the replenishment technique in situations with rapid changes.

10.4 MOTION-COMPENSATED CODING

In addition to the frame-difference predictive coding technique (a variant of the frame replenishment technique discussed above), another technique: displacement-based predictive coding, was developed at almost the same time (Rocca, 1969; Haskell and Limb, 1972a). In this technique, a motion model is assumed. That is, the changes between successive frames are considered due to the translation of moving objects in the image planes. Displacement vectors of objects are first estimated. Differential signals between the intensity value of the picture elements in the moving areas and those of their counterparts in the previous frame, which are translated by the estimated



(a) t_{n-1} (b) t_n

FIGURE 10.6 Two consecutive frames of a video sequence.

displacement, are encoded. This approach, which takes motion into account to compress video sequences, is referred to as motion-compensated predictive coding. It has been found to be much more efficient than the frame-difference prediction technique.

To understand the above statement, let us look at the diagram shown in Figure 10.6. Assume a car translating from the right side to the left side in the image planes at a uniform speed during the time interval between the two consecutive image frames. Other than this, there are no movements or changes in the frames. Under this circumstance, if we know the displacement vector of the car on the image planes during the time interval between two consecutive frames, we can then predict the position of the car in the latter frame from its position in the former frame. One may think that if the translation vector is estimated well, then so is the prediction of the car position. This is true. In reality, however, estimation errors occurring in determination of the motion vector, which may be caused by various noises existing in the frames, may cause the predicted position of the car in the latter frame to differ from the actual position of the car in the latter frame.

The above translational model is a very simple one; it cannot accommodate motions other than translation, say, rotation, and camera zooming. Occlusion and disocclusion of objects make the situation even more complicated since in the occlusion case some portions of the images may disappear, while in the disocclusion case some newly exposed areas may appear. Therefore, the prediction error is almost inevitable. In order to have good-quality frames in the receiver, we can find the prediction error by subtracting the predicted version of the latter frame from the actual version of latter frame. If we encode both the displacement vectors and the prediction error, and transmit these data to the receiver, we may be able to obtain high-quality reconstructed images in the receiver. This is because in the receiving end, using the displacement vectors transmitted from the transmitter and the reconstructed former frame, we can predict the latter frame. Adding the transmitted prediction error to the predicted frame, we may reconstruct the latter frame with satisfactory quality. Furthermore, if manipulating the procedure properly, we are able to achieve data compression.

The displacement vectors are referred to as side or overhead information to indicate their auxiliary nature. It is noted that motion estimation drastically increases the computational complexity of the coding algorithm. In other words, the higher coding efficiency is obtained in motion-compensated coding, but with a higher computational burden. As we pointed out in Section 10.1, this is both technically feasible and economically desired since the cost of digital signal processing decreases much faster than that of transmission (Dubois et al., 1981).

Motion-compensated video compression has become a major development in coding. For more information, readers should refer to several excellent survey papers (Musmann et al., 1985; Zhang et al., 1995; Kunt, 1995).

The common practice of motion-compensated coding in video compression can be split into the following three stages. First, the *motion analysis* stage; that is, displacement vectors for either

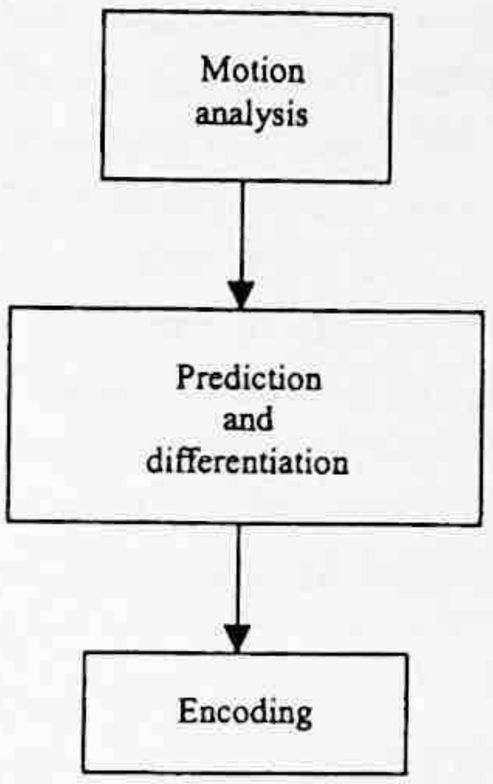


FIGURE 10.7 Block diagram of motion-compensated coding.

every pixel or a set of pixels in image planes from sequential images are estimated. Second, the present frame is predicted by using estimated motion vectors and the previous frame. The prediction error is then calculated. This stage is called *prediction and differentiation*. The third stage is *encoding*. The prediction error (difference between the present and the predicted present frames) and the motion vectors are encoded. Through an appropriate manipulation, the total amount of data for both the motion vectors and prediction error is expected to be much less than the raw data existing in the image frames, thus resulting in data compression. A block diagram of motion-compensated coding is shown in Figure 10.7.

Before leaving this section, we compare the frame replenishment technique with the motion-compensated coding technique. Qualitatively speaking, we see from the above discussion that the replenishment technique is also a kind of predictive coding in nature. This is particularly true if we consider the frame-difference predictive technique used in frame replenishment. There, it uses a pixel's intensity value in the previous frame as an estimator of its intensity value in the present frame.

Now let's look at motion-compensated coding. Consider a pixel on the present frame. Through motion analysis, the motion-compensated technique finds its counterpart in the previous frame. That is, a pixel in the previous frame is identified such that it is supposed to translate to the position on the present frame of the pixel under consideration during the time interval between successive frames. This counterpart's intensity value is used as an estimator of that of the pixel under consideration. We can see that the model used for motion-compensated coding is much more advanced than that used for frame replenishment, therefore, it achieves a much higher coding efficiency. A motion-compensated coding technique that utilized the first pel-recursive algorithm for motion estimation (Netravali and Robbins, 1979) was reported to achieve a bit rate 22 to 50% lower than that obtained by simple frame-difference prediction, a version of frame replenishment.

The more advanced model utilized in motion-compensated coding, on the other hand, leads to higher computational complexity. Consequently, both the coding efficiency and the computational complexity in motion-compensated coding are higher than that in frame replenishment.

10.5 MOTION ANALYSIS

As discussed above, we usually conduct motion analysis in video sequence compression. There, 2-D displacement vectors of a pixel or a group of pixels on image planes are estimated from given image frames. Motion analysis can be viewed from a much broader point of view. It is well known that the vision systems of both humans and animals observe the outside world to ascertain motion and to navigate themselves in the 3-D world space. Two groups of scientists study vision. Scientists

in the first group, including psychophysicists, physicians, and neurophysiologists study human and animal vision. Their goal is to understand biological vision systems — their operation, features, and limitations. Computer scientists and electrical engineers form the second group. As pointed out by Aggarwal and Nandhakumar (1988), their ultimate goal is to develop computer vision systems with the ability to navigate, recognize, and track objects, and estimate their speed and direction. Each group benefits from the research results of the other group. The knowledge and results of research in psychophysics, physiology, and neurophysiology have influenced the design of computer vision systems. Simultaneously, the research results achieved in computer vision have provided a framework in modeling biological vision systems and have helped in remedying faults in biological vision systems. This process will continue to advance research in both groups, hence benefiting society.

10.5.1 Biological Vision Perspective

In the field of biological vision, most scientists consider motion perception as a two-step process, even though there is no ample biological evidence to support this view (Singh, 1991). The two steps are measurement and interpretation. The first step measures the 2-D motion projected on the imaging surfaces. The second step interprets the 2-D motion to induce the 3-D motion and structure on the scene.

10.5.2 Computer Vision Perspective

In the field of computer vision, motion analysis from image sequences is traditionally split into two steps. In the first step, intermediate variables are derived. By *intermediate variables*, we mean 2-D motion parameters in image planes. In the second step, 3-D motion variables, say, speed, displacement, position, and direction, are determined.

Depending on the different intermediate results, all approaches to motion analysis can be basically classified into two categories: feature correspondence and optical flow. In the former category, a few distinct features are first extracted from image frames. For instance, consider an image sequence containing an aircraft. Two consecutive frames are shown in Figure 10.8. The head and tail of the aircraft, and the tips of its wings may be chosen as features. The correspondence of these features on successive image frames needs to be established. In the second step, 3-D motion can then be analyzed from the extracted features and their correspondence in successive frames. In the latter category of approaches, the intermediate variables are optical flow. An optical flow vector is defined as a velocity vector of a pixel on an image frame. An optical flow field is referred to as the collection of the velocity vectors of all the pixels on the frame. In the first step, optical flow vectors are determined from image sequences as the intermediate variables. In the second step, 3-D motion is estimated from optical flow. It is noted that optical flow vectors are closely

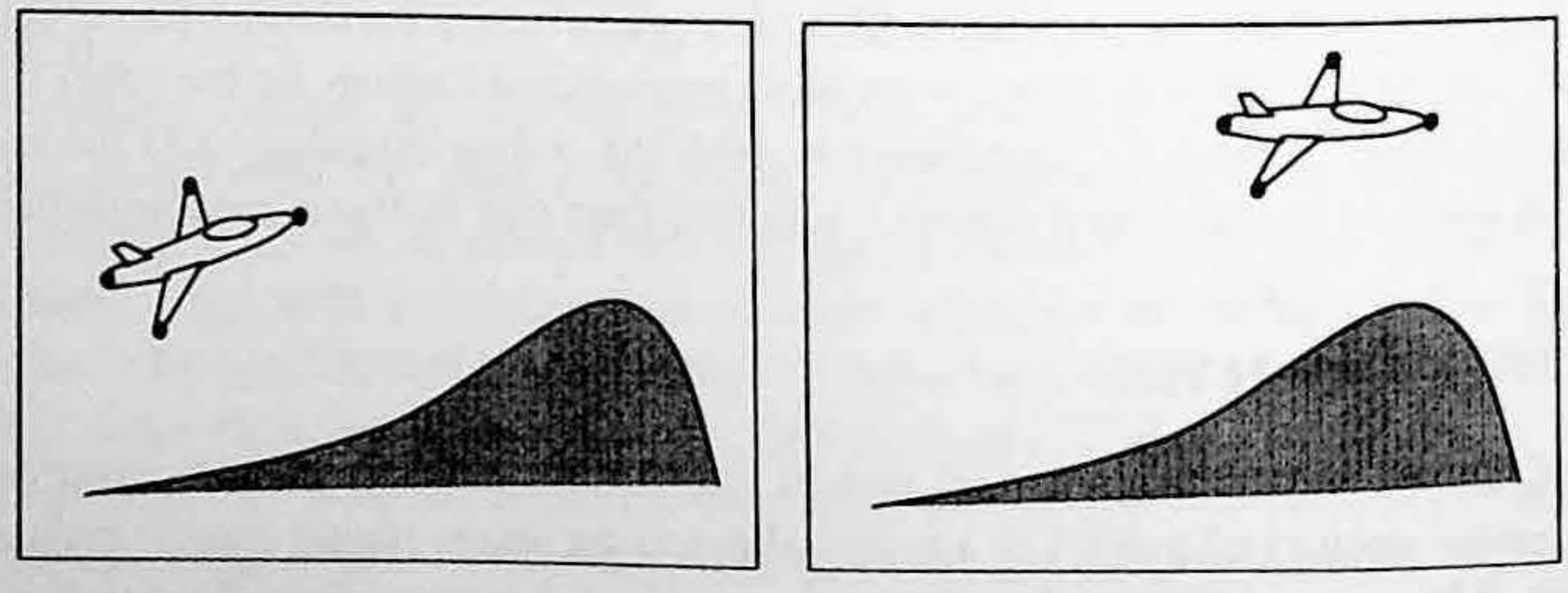


FIGURE 10.8 Feature extraction and correspondence from two consecutive frames in a temporal image sequence.

related to displacement vectors in that a velocity vector multiplying by the time interval between two consecutive frames results in the corresponding displacement vector. Optical flow and its determination will be discussed in detail in Chapter 13.

It is noted that there is a so-called direct method in motion analysis. Contrary to the above optical flow approach, instead of determining 2-D motion variables, (i.e., the intermediate variables), prior to 3-D motion estimation, the direct method attempts to estimate 3-D motion without explicitly solving for the intermediate variables. In (Huang and Tsai, 1981b) the equation characterizing displacement vectors in the 2-D image plane and the equation characterizing motion parameters in 3-D world space are combined so that the motion parameters in 3-D world space can be directly derived. This method has been utilized to recover structure (object surfaces) in 3-D world space as well (Negahdaripour and Horn, 1987; Horn and Weldon, 1988; Shu and Shi, 1993). The direct method has certain limitations. That is, if the geometry of object surfaces is not known in advance, then the method fails.

The feature correspondence approach is sometimes referred to as the discrete approach, while the optical flow approach is sometimes referred to as the continuous approach. This is because the correspondence approach concerns only a set of relatively sparse but highly discriminatory 2-D features on image planes. The optical flow approach is concerned with a dense field of motion vectors.

It has been found that both feature extraction and correspondence establishment are not trivial tasks. Occlusion and disocclusion which, respectively, cause some features to disappear and some features to reappear, make feature correspondence even more difficult. The development of robust techniques to solve the correspondence problem is an active research area and is still in its infancy. So far, only partial solutions suitable for simplistic situations have been developed (Aggarwal and Nandhakumar, 1988). Hence the feature correspondence approach is rarely used in video compression. Because of this, we will not discuss this approach any further.

Motion analysis (sometimes referred to as motion estimation or motion interpretation) from image sequences is necessary in automated navigation. It has played a central role in the field of computer vision since the late 1970s and early 1980s. A great deal of the papers presented at the International Conference on Computer Vision cover this and related topics. Many workshops, symposiums, and special sessions are organized around this subject (Thompson, 1989).

10.5.3 Signal Processing Perspective

In the field of signal processing, motion analysis is mainly considered in the context of bandwidth reduction and/or data compression in the transmission of visual signals. Therefore, instead of the motion in 3-D world space, only the 2-D motion in the image plane is concerned.

Because of the real-time nature in visual transmission, the motion model cannot be very complicated. So far, the 2-D translational model is most frequently assumed in the field. In the 2-D translational model it is assumed that the change between a frame and its previous one is due to the motion of objects in the frame plane during the time interval between two consecutive frames. In many cases, as long as frames are taken densely enough, this assumption is valid. By *motion analysis* we mean the estimation of translational motion — either the displacement vectors or velocity vectors. With this kind of motion analysis, one can apply the motion-compensated coding discussed above, making coding more efficient.

Basically there are three techniques in 2-D motion analysis: correlation, and recursive and differential techniques. Philosophically speaking, the first two techniques belong to the same group: region matching.

Refer to Figure 10.6, where the moving car is the object under investigation. By *motion analysis* we mean finding the displacement vector, i.e., a vector representing the relative positions of the car in the two consecutive frames. With region matching, one may consider the car (or a portion of the car) as a region of interest, and seek the best match between the two regions in the two

frames: specifically, the region in the present frame and the region in the previous frame. For identifying the best match, two techniques, the correlation and the recursive methods, differ in methodology. The correlation technique finds the best match by searching the maximum correlation between the two regions in a predefined search range, while the recursive technique estimates the best match by recursively minimizing a nonlinear measurement of the dissimilarities between the two regions.

A couple of comments are in order. First, it is noted that the most frequently used technique in motion analysis is called block matching, which is a type of the correlation technique. There, a video frame is divided into nonoverlapped rectangular blocks with each block having the same size, usually 16 × 16. Each block thus generated is assumed to move as one, i.e., all pixels in a block share the same displacement vector. For each block, we find its best match in the previous frame with correlation. That is, the block in the previous frame, which gives the maximum correlation, is identified. The relative position of these two best matched blocks produces a displacement vector. This block matching technique is simple and very efficient, and will be discussed in detail in Chapter 11. Second, as multimedia finds more and more applications, the regions occupied by arbitrarily-shaped objects (no longer always rectangular blocks) become increasingly important in content-based video retrieval and manipulation. Motion analysis in this case is discussed in Chapter 18. Third, although the recursive technique is categorized as a region matching technique, it may be used for finding displacement vectors for individual pixels. In fact the recursive technique was originally developed for determining displacement vectors of pixels and, hence, it is called pel-recursive. This technique is discussed in Chapter 12. Fourth, both correlation and recursive techniques can be utilized for determining optical flow vectors. Optical flow is discussed in Chapter 13.

The third technique in 2-D motion analysis is the differential technique. This is one of the main techniques utilized in determining optical flow vectors. It is named after the term of differentials because it uses partial differentiation of an intensity function with respect to the spatial coordinates x and y, as well as the temporal coordinate t. This technique is also discussed in Chapter 13.

10.6 MOTION COMPENSATION FOR IMAGE SEQUENCE PROCESSING

Motion analysis has long been considered a key issue in image sequence processing (Huang, 1981a; Shi, 1997). Obviously, in an area like automated navigation, motion analysis plays a central role. From the discussion in this chapter, we see that motion analysis also plays a key role in video data compression. Specifically, we have discussed the concept of motion-compensated video coding in Section 10.4. In this section we would like to consider motion compensation for image sequence processing, in general. Let us first consider motion-compensated interpolation. Then, we will discuss motion-compensated enhancement, restoration, and down-conversion.

10.6.1 Motion-Compensated Interpolation

Interpolation is a simple yet efficient and important method in image and video compression. In image compression, we may only transmit, say, every other row. We then try to interpolate these missing rows from the other half of the transmitted rows in the receiver. In this way, we compress the data to half. Since the interpolation is carried out within a frame, it is referred to as *spatial* interpolation. In video compression, for instance, in videophone service, instead of transmitting 30 frames per second, we may choose a lower frame rate, say, 10 frames per second. In the receiver, we may try to interpolate the dropped frames from the transmitted frames. This strategy immediately drops the transmitted data to one third. Another example is the conversion of a motion picture into an NTSC (National Television System Commission) TV signal. There, every first frame in the

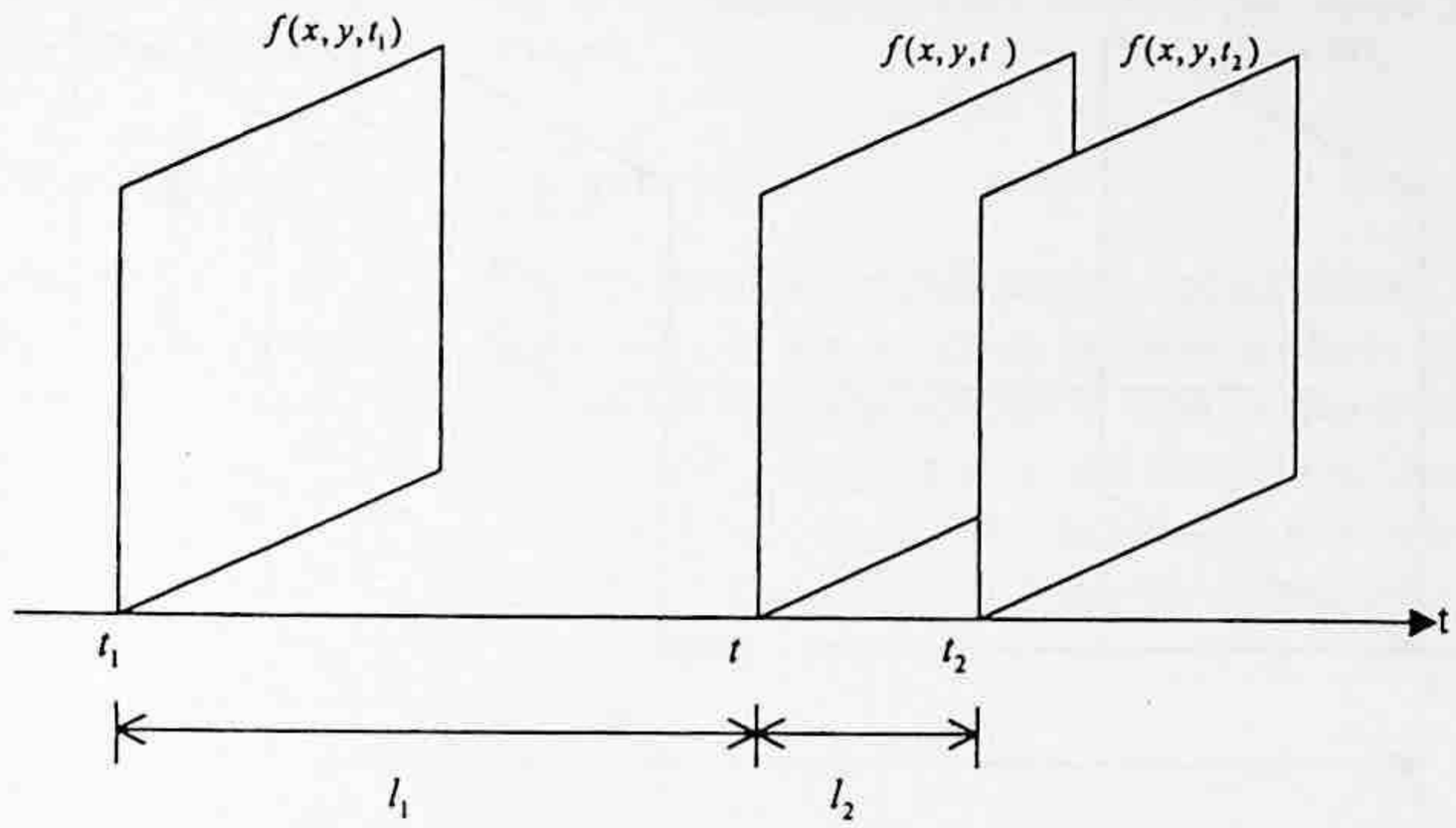


FIGURE 10.9 Weighted linear interpolation.

motion picture is repeated three times and the next frame twice, thus converting a 24-frame-per-second motion picture to a 60-field-per-second NTSC signal. This is commonly referred to as 3:2 pulldown. In these two examples concerning video, interpolation is along the temporal dimension, which is referred to as *temporal* interpolation.

For basic concepts of zero-order interpolation, bilinear interpolation, and polynomial interpolation, readers are referred to signal processing texts, for instance (Lim, 1990). In temporal interpolation, the zero-order interpolation means creation of a frame by copying its nearest frame along the time dimension. The conversion of a 24-frame-per-second motion picture to a 60-field-per-second NTSC signal can be classified into this type of interpolation. Weighted linear interpolation can be illustrated with Figure 10.9.

There, the weights are determined according to the lengths of time intervals, which is similar to the bilinear interpolation widely used in spatial interpolation, except that here only one index (along the time axes) is used, while two indexes (along two spatial axes) are used in spatial bilinear interpolation. That is,

$$f(x,y,t)=\frac{l_2}{l_1+l_2}f(x,y,t_1)+\frac{l_1}{l_1+l_2}f(x,y,t_2) \tag{10.4}$$

If there are one or multiple moving objects existing in successive frames, however, the weighted linear interpolation will blur the interpolated frames. Taking motion into account in the interpolation results in motion-compensated interpolation. In Figure 10.10, we still use the three frames shown in Figure 10.9 to illustrate the concept of motion-compensated interpolation. First, motion between two given frames is estimated. That is, the displacement vectors for each pixel are determined. Second, we choose a frame that is nearer to the frame we want to interpolate. Third, the displacement vectors determined in the first step are proportionally converted to the frame to be created. Each pixel in this frame is projected via the determined motion trajectory to the frame chosen in step 2. In the process of motion-compensated interpolation, spatial interpolation in the frame chosen in step 2 usually is needed.

10.6.2 Motion-Compensated Enhancement

It is well known that when an image is corrupted by additive white Gaussian noise (AWGN) or burst noise, linear low-pass filtering, such as simple averaging or nonlinear low-pass filtering, such as a median filter, performs well in removing the noise. When an image sequence is concerned,

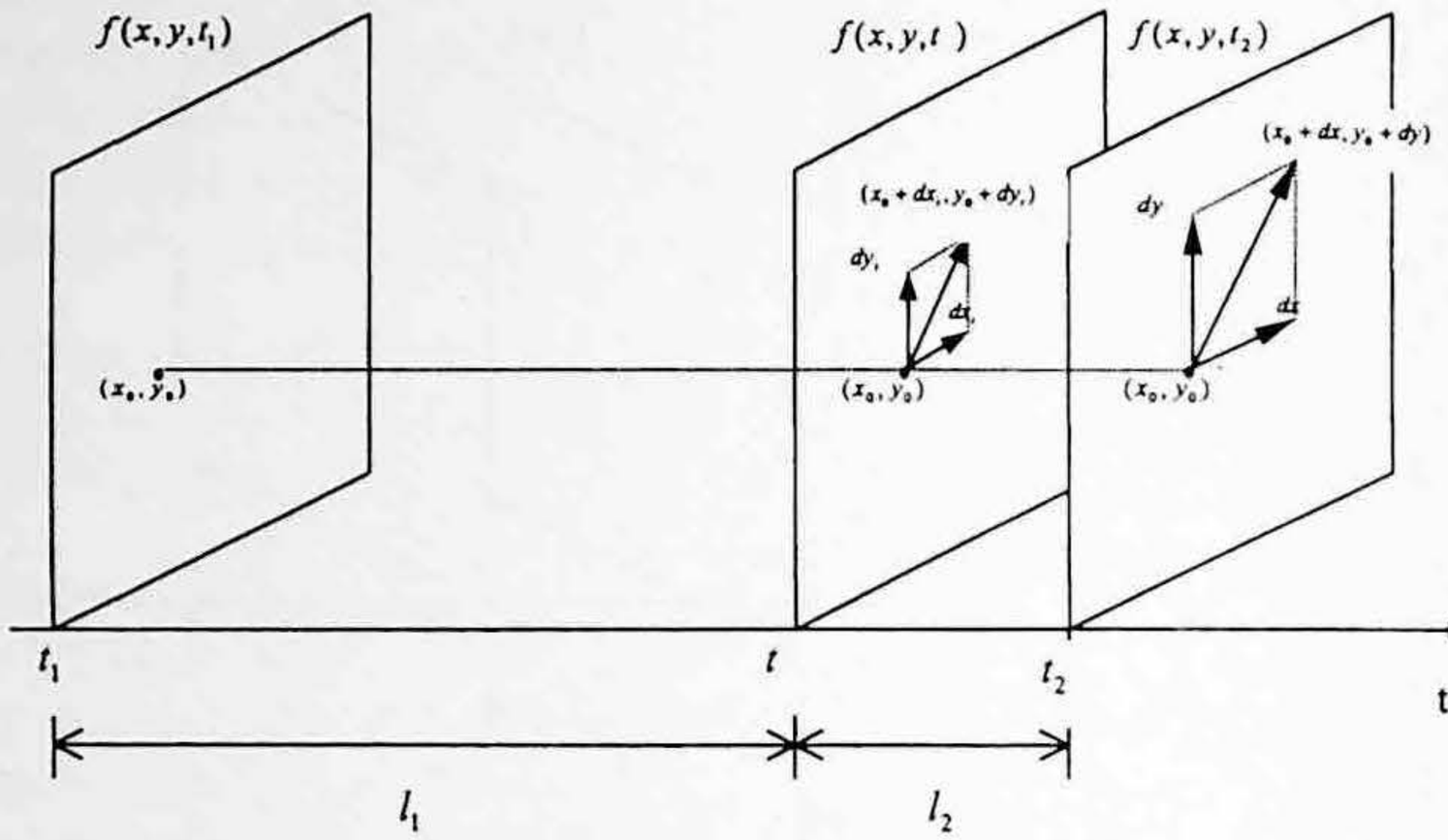


FIGURE 10.10 Motion-compensated interpolation.

we may apply such types of filtering along the temporal dimension to remove noise. This is called temporal filtering. These types of low-pass filtering may blur images, an effect that may become quite serious when motion exists in image planes. The enhancement, which takes motion into account, is referred to as motion-compensated enhancement, and has been found very efficient in temporal filtering (Huang and Hsu, 1981c).

To facilitate the discussion, we consider simple averaging as a means for noise filtering in what follows. It is understood that other filtering techniques are possible, and that everything discussed here is applicable there. Instead of simply averaging n successive image frames in a video sequence, motion-compensated temporal filtering will first analyze the motion existing in these frames. That is, we estimate the motion of pixels in successive frames first. Then averaging will be conducted only on those pixels along the same motion trajectory. In Figure 10.11, three successive frames are shown and denoted by $f(x, y, t_1)$, $f(x, y, t_2)$, and $f(x, y, t_3)$, respectively. Assume that three pixels, denoted by (x_1, y_1), (x_2, y_2), and (x_3, y_3), respectively, are identified to be perspective projections of the same object point in the 3-D world space on the three frames. The averaging is then applied to these three pixels. It is noted that the number of successive frames, n, may not necessarily have to be three. Motion analysis can use any one of the several techniques discussed in Section 10.5.

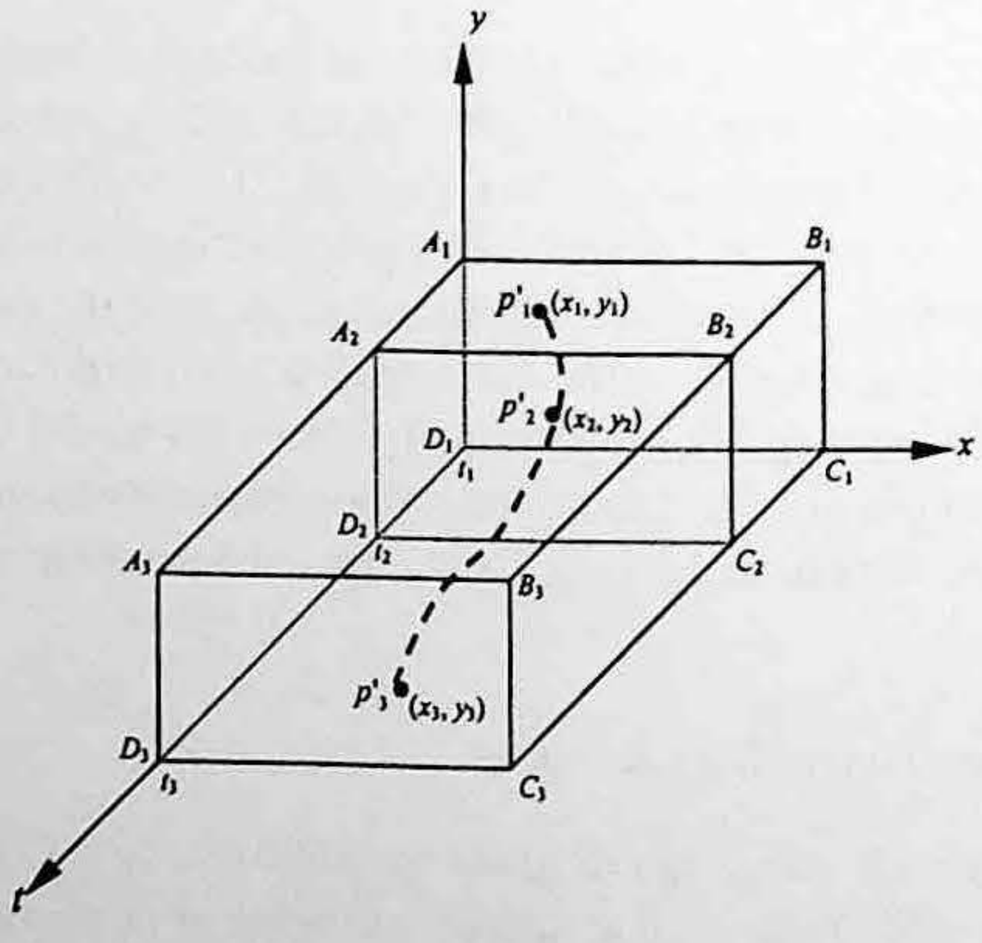


FIGURE 10.11 Motion-compensated temporal filtering.

Motion-compensated temporal filtering is not necessarily implemented pixelwise; it can also be used objectwise, or regionwise.

10.6.3 Motion-Compensated Restoration

Extensive attention has been paid to the restoration of full-length feature films. There, typical artifacts are due to dirt and sparkle. Early work in the detection of these artifacts ignored motion information completely. Late motion estimation has been utilized to detect these artifacts based on the assumption that the artifacts occur occasionally along the temporal dimension. Once the artifacts have been found, motion-compensated temporal filtering and/or interpolation will be used to remove the artifacts. One successful algorithm for the detection and removal of anomalies in digitized animation film can be found in (Tom et al., 1998).

10.6.4 Motion-Compensated Down-Conversion

Here we present one more example in which motion compensation finds application in digital video processing.

It is believed that there will be a need to down-convert a high definition television (HDTV) image sequence for display onto an NTSC monitor during the upcoming transition to digital television broadcast. The most straightforward approach is to fully decode the image sequence first, then apply a prefiltering and subsampling process to each field of the interlaced sequence. This is referred to as a full-resolution decoder (FRD). The merit of this approach is the high quality achieved, while the drawback is a high cost in terms of the large amount of memory required to store the reference frames. To reduce the required memory space, another approach is considered. In this approach, the down-conversion is conducted within the decoding loop and is referred to as a low-resolution decoder (LRD). It can significantly reduce the required memory and still achieve a reasonably good picture quality.

The prediction drift is a major type of artifact existing in the down-conversion. It is defined as the successive blurring of forward-predicted frames with a group of pictures. It is caused mainly by non-ideal interpolation of sub-pixel intensities and the loss of high-frequency data within the block. An optimal set of filters to perform low-resolution motion compensation has been derived to effectively minimize the drift. For details on an algorithm in the down-conversion utilizing an optimal motion compensation scheme, readers are referred to Vetro and Sun (1998).

10.7 SUMMARY

After Section II, still image compression, we shift our attention to video compression. Prior to Section IV, where we discuss various video compression algorithms and standards, however, we first address the issue of motion analysis and motion compensation in this chapter that starts Section III, motion estimation and compensation. This is because video compression has its own characteristics, which are different from those of still image compression. The main difference lies in interframe correlation.

In this chapter, the concept of various image sequences is discussed in a broad scope. In doing so, a single image, temporal image sequences, and spatial image sequences are all unified under the concept of imaging space. The redundancy between pixels in successive frames is analyzed for both videoconferencing and TV broadcast cases. In these applications, there is more interframe correlation than intraframe correlation, in general. Therefore, the utilization of interframe correlation becomes a key issue in video compression.

There are two major techniques in exploitation of interframe correlation: frame replenishment and motion compensation. In the conditional replenishment technique, only those pixel gray level values, whose variation from their counterparts in the previous frame exceeds a threshold, are

encoded and transmitted to the receiver. These pixels are called changing pixels. For pixels other than the changing pixels, their gray values are just repeated in the receiver. This simplest frame replenishment technique achieves higher coding efficiency than coding each pixel in each frame due to the utilization of interframe redundancy. In the more advanced frame replenishment techniques, say, the frame-difference predictive coding technique, both temporal and spatial neighboring gray values of the pixels are used to predict that of a changing pixel. Instead of the intensity values of the changing pixels, the prediction error is encoded and transmitted. Because the variance of the prediction error is smaller than that of the intensity values, this more advanced frame replenishment technique is more efficient than the conditional replenishment technique.

The main drawback of frame replenishment techniques is associated with rapid motion and/or intensity variation occurring on the image planes. Under these circumstances, frame replenishment will suffer from the dirty window effect, and even buffer saturation.

In motion-compensated coding, the motion of pixels is first analyzed. Based on the previous frame and the estimated motion, the current frame is predicted. The prediction error together with motion vectors are encoded and transmitted to the receiver. Due to more accurate prediction based on a motion model, motion-compensated coding achieves higher coding efficiency compared with frame replenishment. This is conceivable because frame replenishment basically uses the intensity value of a pixel in the previous frame to predict that of the pixel in the same location in the present frame, while the prediction in motion-compensated coding uses motion trajectory. This implies that higher coding efficiency is obtained in motion compensation at the cost of higher computational complexity. This is technically feasible and economically desired since the cost of digital signal processing decreases much faster than that of transmission.

Because of the real-time requirement in video coding, only a simple 2-D translational model is used. There are mainly three types of motion analysis techniques used in motion-compensated coding. They are block matching, pel-recursion, and optical flow. By far, block matching is used most frequently. These three techniques are discussed in detail in the following three chapters.

Motion compensation is also widely utilized in other tasks of digital video sequence processing. Examples include motion-compensated interpolation, motion-compensated enhancement, motion-compensated restoration, and motion-compensated down-conversion.

10.8 EXERCISES

10-1. Explain the analogy between a stereo image sequence vs. the imaging space, and a stereo image pair vs. the spatial image sequence to which the stereo image pair belongs.

10-2. Explain why the imaging space can be considered as a unification of image frames, spatial image sequences, and temporal image sequences.

10-3. Give the definitions of the following several concepts: image, image sequence, and video. Discuss the relationship between them.

10-4. What feature causes video compression to be quite different from still image compression?

10-5. Describe the conditional replenishment technique. Why can it achieve higher coding efficiency in video coding than those techniques encoding each pixel in each frame?

10-6. Describe the frame-difference predictive coding technique. You may want to refer to Section 3.5.2.

10-7. What is the main drawback of frame replenishment?

10-8. Both the frame-difference predictive coding and motion-compensated coding are predictive codings in nature.

(a) What is the main difference between the two?

(b) Explain why motion-compensated coding is usually more efficient.

(c) What is the price paid for higher coding efficiency with motion-compensated coding?

10-9. Motion analysis is an important task encountered in both computer vision and video coding. What is the major different requirement for motion analysis in these two fields?

10-10. Work on the first 40 frames of a video sequence other than the Miss America sequence. Determine, on an average basis, what percentage of the total pixels change their gray-level values by more than 1% of the peak signal between two consecutive frames.

10-11. Similar to the experiment associated with Figure 10.5, do your own experiment to observe the dirty window effect. That is, work on two successive frames of a video sequence chosen by yourself, and only update a part of those changing pixels.

10-12. Take two frames from the Miss America sequence or from another sequence of your own choice in which a relatively large amount of motion is involved.

(a) Using the weighted linear interpolation defined in Equation 10.4, create an interpolated frame, which is located in the 1/3 of the time interval from the second frame (i.e., $l_2 = \frac{1}{3}(l_1 + l_2)$ according to Figure 10.9).

(b) Using motion-compensated interpolation, create an interpolated frame at the same position along the temporal dimension.

(c) Compare the two interpolated frames and make your comments.

REFERENCES

Aggarwal, J. K. and N. Nandhakumar, On the computation of motion from sequences of images — a review, *Proc. IEEE,* 76(8), 917-935, 1988.

Dubois, E., B. Prasada and M. S. Sabri, Image Sequence Coding, in *Image Sequence Analysis*, T. S. Huang, Ed., Springer-Verlag, New York, 1981, chap. 3.

Haskell, B. G. and J. O. Limb, Predictive Video Encoding Using Measured Subject Velocity, U.S. Patent 3,632, 865, January 1972.

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

Haskell, B. G. Frame Replenishment Coding of Television, in *Image Transmission Techniques*, W. K. Pratt, Ed., Academic Press, New York, 1979, chap. 6.

Horn, B. K. P. and B. G. Schunck, Determining optical flow, *Artificial Intelligence*, 17, 185-203, 1981.

Horn, B. K. P. and E. J. Weldon, Jr., Direct methods for recovering motion, *Int. J. Comput. Vision,* 2, 51-76, 1988.

Huang, T. S. Ed., *Image Sequence Analysis*, Springer-Verlag, New York, 1981.

Huang, T. S. and R. Y. Tsai, Image Sequence Analysis: Motion Estimation, in *Image Sequence Analysis*, T. S. Huang, Ed., Springer-Verlag, New York, 1981.

Huang, T. S. and Y. P. Hsu, Image Sequence Enhancement, in *Image Sequence Analysis*, T. S. Huang, Ed., Springer-Verlag, New York, 1981.

Huang, T. S., Ed., *Image Sequence Processing and Dynamic Scene Analysis*, Springer-Verlag, New York, 1983.

Jain, A. K. *Fundamentals of Digital Image Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1989.

Kretzmer, E. R. Statistics of television signal, *Bell Syst. Tech. J.,* 31(4), 751-763, 1952.

Kunt, M., Ed., Special Issue on Digital Television. Part I: Technologies, *Proc. IEEE,* 83(6), 1995.

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), 2545-1554, 1969.

Musmann, H. G., P. Pirsch, and H. J. Grallert, Advances in picture coding, *Proc. IEEE,* 73(4), 523-548, 1985.

Negahdaripour, S. and B. K. P. Horn, Direct passive navigation, *IEEE Trans. Pattern Anal. Machine Intell.,* PAMI-9(1), 168-176, 1987.

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

Pratt, W. K., Ed., *Image Transmission Techniques*, Academic Press, New York, 1979.

Rocca, F. Television bandwidth compression utilizing frame-to-frame correlation and movement compensation, *Symposium on Picture Bandwidth Compression*, Gordon and Breach, Newark, NJ, 1972.

Seyler, A. J. The coding of visual signals to reduce channel-capacity requirements, *IEEE Monogr.* 533E, July 1962.

Seyler, A. J. Probability distributions of television frame difference, *Proc. IREE (Australia)*, 26, 335, 1965.
Singh, A. *Optical Flow Computation: A Unified Perspective*, IEEE Press, New York, 1991.
Shi, Y. Q., C. Q. Shu, and J. N. Pan, Unified optical flow field approach to motion analysis from a sequence of stereo images, *Pattern Recognition*, 27(12), 1577-1590, 1994.
Shi, Y. Q. Editorial introduction to special issue on image sequence processing, *Int. J. Imaging Syst. Technol.*, 9(4), 189-191, 1998.
Shu, C. Q. and Y. Q. Shi, On unified optical flow field, *Pattern Recognition*, 24(6), 579-586, 1991.
Shu, C. Q. and Y. Q. Shi, Direct recovering of Nth order surface structure using unified optical flow field, *Pattern Recognition*, 26(8), 1137-1148, 1993.
Tekalp, A. M. *Digital Video Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1995.
Thompson, W. B. Introduction to special issue on visual motion, *IEEE Trans. Pattern Anal. Machine Intell.*, 11(5), 449-450, 1989.
Tom, B. C., M. G. Kang, M. C. Hong and A. K. Katsaggelos, Detection and removal of anomalies in digitized animation film, *Int. J. Imaging Syst. Technol.*, 9(4), 283-293, 1998.
Vetro, A. and H. Sun, Frequency domain down-conversion of HDTV using an optimal motion compensation scheme, *Int. J. Imaging Syst. Technol.*, 9(4), 274-282, 1998.
Waxman, A. M. and J. H. Duncan, Binocular image flow: steps towards stereo-motion fusion, *IEEE Trans. Pattern Anal. Machine Intell.*, PAMI-8(6), 715-729, 1986.
Westwater, R. and B. Furht, *Real-Time Video Compression*, Kluwer Academic, Norwall, MA, 1997.
Zhang, Y.-Q., W. Li, and M. L. Liou, Ed., Special Issue on Advances in Image and Video Compression, *Proc. IEEE*, 83(2), 1995.

11 Block Matching

As mentioned in the previous chapter, displacement vector measurement and its usage in motion compensation in interframe coding for a TV signal can be traced back to the 1970s. Netravali and Robbins (1979) developed a pel-recursive technique, which estimates the displacement vector for each pixel recursively from its neighboring pixels using an optimization method. Limb and Murphy (1975), Rocca and Zanoletti (1972), Cafforio and Rocca (1976), and Brofferio and Rocca (1977) developed techniques for the estimation of displacement vectors of a block of pixels. In the latter approach, an image is first segmented into areas with each having an approximately uniform translation. Then the motion vector is estimated for each area. The segmentation and motion estimation associated with these arbitrarily shaped blocks are very difficult. When there are multiple moving areas in images, the situation becomes more challenging. In addition to motion vectors, the shape information of these areas needs to be coded. Hence, when moving areas have various complicated shapes, both computational complexity and coding load will increase remarkably.

In contrast, the block matching technique, which is the focus of this chapter, is simple, straightforward, and yet very efficient. It has been by far the most popularly utilized motion estimation technique in video coding. In fact, it has been adopted by all the international video coding standards: ISO, MPEG-1 and MPEG-2, and ITU H.261, and H.263. These standards will be introduced in detail in Chapters 16, 17, and 19, respectively.

It is interesting to note that even nowadays, with the tremendous advancements in multimedia engineering, object-based and/or content-based manipulation of audiovisual information is still very demanding, particularly in audiovisual data storage, retrieval, and distribution. The applications include digital library, video on demand, audiovisual databases, and so on. Therefore, the coding of arbitrarily shaped objects has attracted great research attention these days. It has been included in the MPEG-4 activities (Brailean, 1997), and will be discussed in Chapter 18.

In this chapter various aspects of block matching are addressed. They include the concept and algorithm, matching criteria, searching strategies, limitations, and new improvements.

11.1 NONOVERLAPPED, EQUALLY SPACED, FIXED SIZE, SMALL RECTANGULAR BLOCK MATCHING

To avoid the kind of difficulties encountered in motion estimation and motion compensation with arbitrarily shaped blocks, the block matching technique was proposed by Jain and Jain (1981) based on the following simple motion model.

An image is partitioned into a set of nonoverlapped, equally spaced, fixed size, small rectangular blocks; and the translation motion within each block is assumed to be uniform. Although this simple model considers translation motion only, other types of motions, such as rotation and zooming of large objects, may be closely approximated by the piecewise translation of these small blocks provided that these blocks are small enough. This observation, originally made by Jain and Jain, has been confirmed again and again since then.

Displacement vectors for these blocks are estimated by finding their best matched counterparts in the previous frame. In this manner, motion estimation is significantly easier than that for arbitrarily shaped blocks. Since the motion of each block is described by only one displacement vector, the side information on motion vectors decreases. Furthermore, the rectangular shape

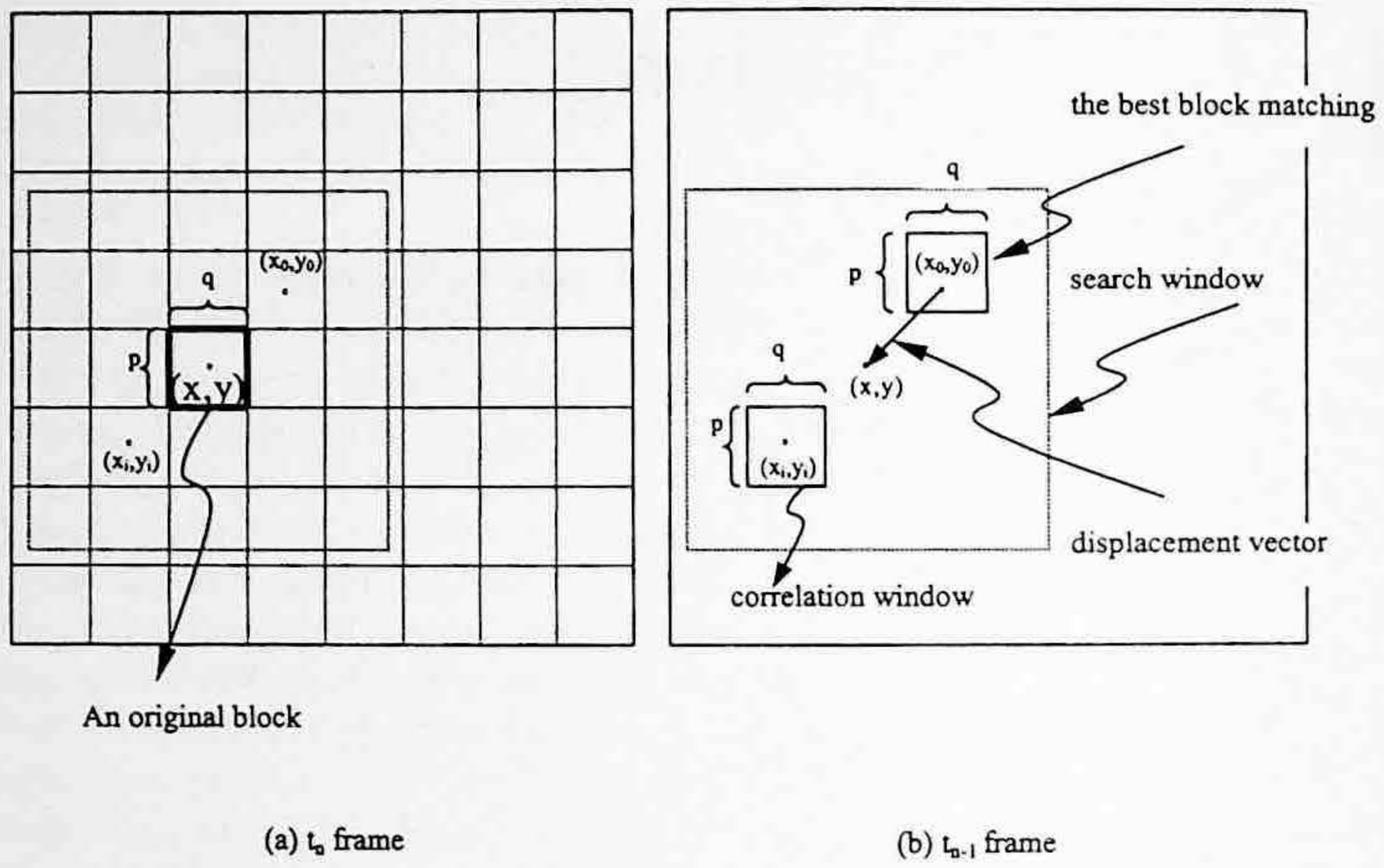


FIGURE 11.1 Block matching.

information is known to both the encoder and the decoder, and hence does not need to be encoded, which saves both computation load and side information.

The block size needs to be chosen properly. In general, the smaller the block size, the more accurate is the approximation. It is apparent, however, that the smaller block size leads to more motion vectors being estimated and encoded, which means an increase in both computation and side information. As a compromise, a size of 16×16 is considered to be a good choice. (This has been specified in international video coding standards such as H.261, H.263, and MPEG-1 and MPEG-2.) Note that for finer estimation a block size of 8×8 is sometimes used.

Figure 11.1 is utilized to illustrate the block matching technique. In Figure 11.1(a) an image frame at moment t_n is segmented into nonoverlapped $p \times q$ rectangular blocks. As mentioned above, in common practice, square blocks of $p = q = 16$ are used most often. Consider one of the blocks centered at (x, y). It is assumed that the block is translated as a whole. Consequently, only one displacement vector needs to be estimated for this block. Figure 11.1(b) shows the previous frame: the frame at moment t_{n-1}. In order to estimate the displacement vector, a rectangular search window is opened in the frame t_{n-1} and centered at the pixel (x, y). Consider a pixel in the search window, a rectangular correlation window of the same size $p \times q$ is opened with the pixel located in its center. A certain type of similarity measure (correlation) is calculated. After this matching process has been completed for all candidate pixels in the search window, the correlation window corresponding to the largest similarity becomes the best match of the block under consideration in frame t_n. The relative position between these two blocks (the block and its best match) gives the displacement vector. This is shown in Figure 11.1(b).

The size of the search window is determined by the size of the correlation window and the maximum possible displacement along four directions: upward, downward, rightward, and leftward. In Figure 11.2 these four quantities are assumed to be the same and are denoted by d. Note that d is estimated from *a priori* knowledge about the translation motion, which includes the largest possible motion speed and the temporal interval between two consecutive frames, i.e., $t_n - t_{n-1}$.

11.2 MATCHING CRITERIA

Block matching belongs to image matching and can be viewed from a wider perspective. In many image processing tasks, we need to examine two images or two portions of images on a pixel-by-pixel

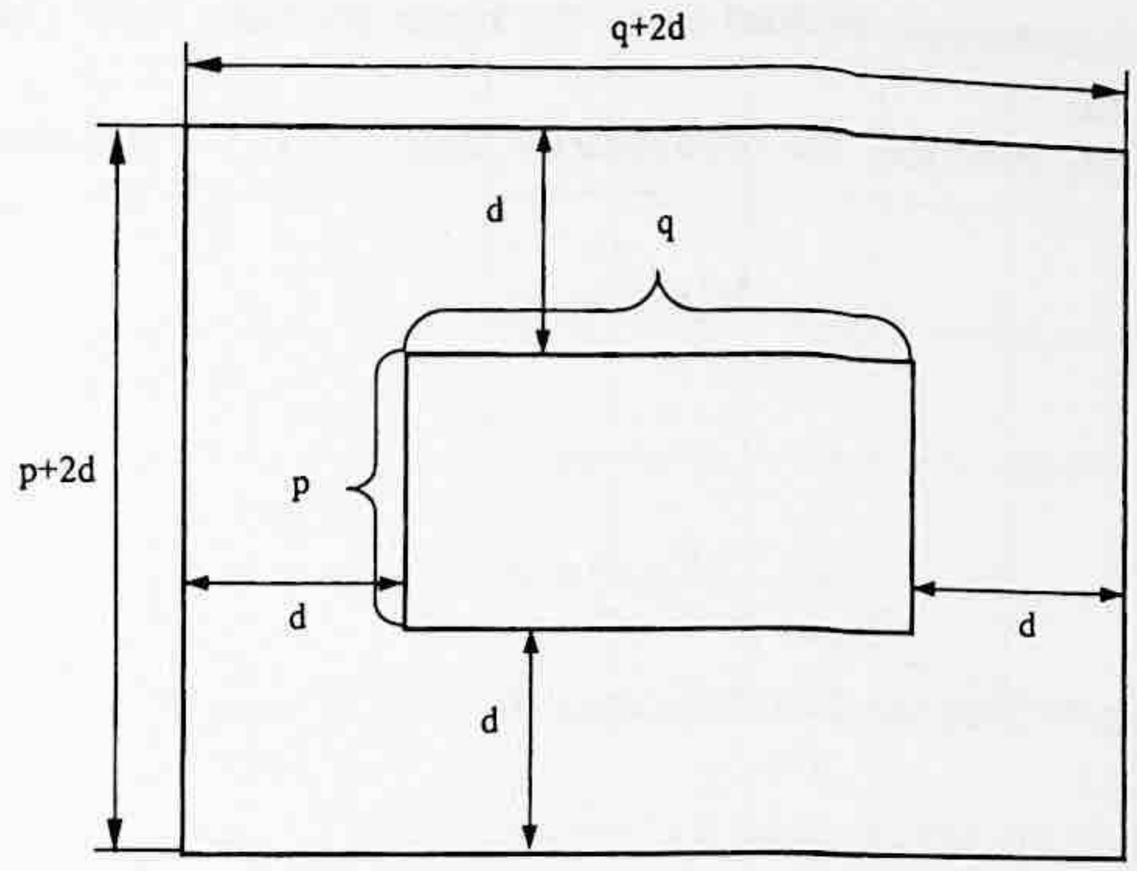


FIGURE 11.2 Search window and correlation window.

basis. These two images or two image regions can be selected from a spatial image sequence, i.e., from two frames taken at the same time with two different sensors aiming at the same object, or from a temporal image sequence, i.e., from two frames taken at two different moments by the same sensor. The purpose of the examination is to determine the similarity between the two images or two portions of images. Examples of this type of application include image registration (Pratt, 1974) and template matching (Jain, 1989). The former deals with spatial registration of images, while the latter extracts and/or recognizes an object in an image by matching the object template and a certain area of the image.

The similarity measure, or correlation measure, is a key element in the matching process. The basic correlation measure between two images t_n and t_{n-1}, C (s, t), is defined as follows (Anuta, 1969).

$$C(s,t)=\frac{\sum_{j=1}^{p}\sum_{k=1}^{q} f_n(j,k)f_{n-1}(j+s,k+t)}{\sqrt{\sum_{j=1}^{p}\sum_{k=1}^{q} f_n(j,k)^2}\sqrt{\sum_{j=1}^{p}\sum_{k=1}^{q} f_{n-1}(j+s,k+t)^2}}. \tag{11.1}$$

This is also referred to as a normalized two-dimensional cross-correlation function (Musmann et al., 1985).

Instead of finding the maximum similarity or correlation, an equivalent but yet more computationally efficient way of block matching is to find the minimum dissimilarity, or matching error. The dissimilarity (sometimes referred to as the error, distortion, or distance) between two images t_n and t_{n-1}, D (s, t) is defined as follows.

$$D(s,t)=\frac{1}{lm}\sum_{j=1}^{p}\sum_{k=1}^{q} M\left(f_n(j,k), f_{n-1}(j+s,k+t)\right), \tag{11.2}$$

where M(u,v) is a metric that measures the dissimilarity between the two arguments u and v. The D (s, t) is also referred to as the matching criterion or the D values.

In the literature there are several types of matching criteria, among which the mean square error (MSE) (Jain and Jain, 1981) and mean absolute difference (MAD) (Koga et al., 1981) are used most often. It is noted that the sum of the squared difference (SSD) (Anandan, 1987) or the sum of the squared error (SSE) (Chan et al., 1990) is essentially the same as MSE. The mean

absolute difference is sometimes referred to as the mean absolute error (MAE) in the literature (Nogaki and Ohta, 1972).

In the MSE matching criterion, the dissimilarity metric M (u, v) is defined as

$$M(u,v) = (u-v)^2. \tag{11.3}$$

In the MAD,

$$M(u,v) = |u-v|. \tag{11.4}$$

Obviously, both criteria are simpler than the normalized two-dimensional cross-correlation measure defined in Equation 11.1.

Before proceeding to the next section, a comment on the selection of the dissimilarity measure is due. A study based on experimental works reported that the matching criterion does not significantly affect the search (Srinivasan, 1984). Hence, the MAD is preferred due to its simplicity in implementation (Musmann et al., 1985).

11.3 SEARCHING PROCEDURES

The searching strategy is another important issue to deal with in block matching. Several searching strategies are discuused below.

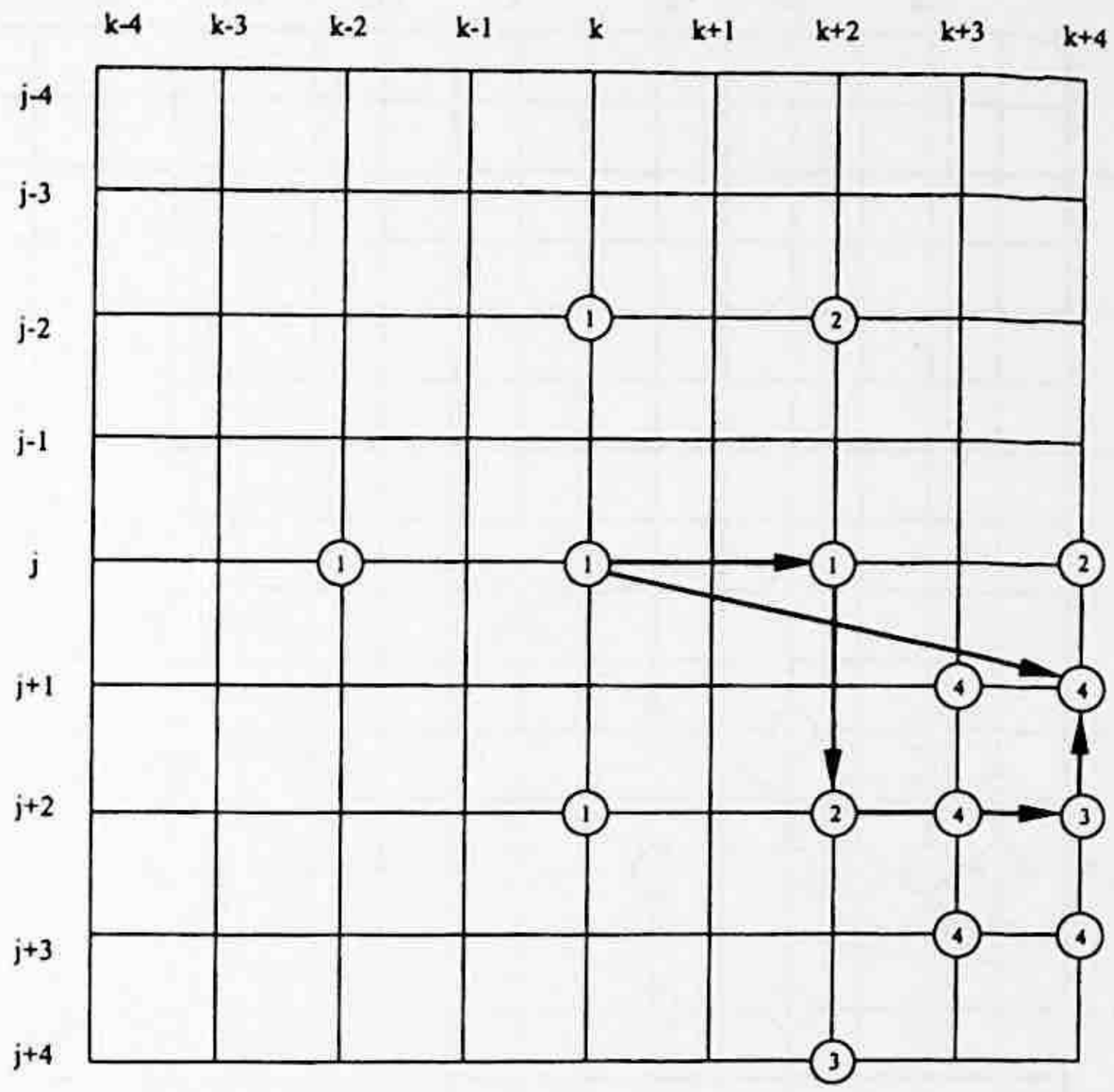
11.3.1 Full Search

Figure 11.2 shows a search window, a correlation window, and their sizes. In searching for the best match, the correlation window is moved to each candidate position within the search window. That is, there are a total (2 d+1) × (2 d+1) positions that need to be examined. The minimum dissimilarity gives the best match. Apparently, this full search procedure is brute force in nature. While the full search delivers good accuracy in searching for the best match (thus, good accuracy in motion estimation), a large amount of computation is involved.

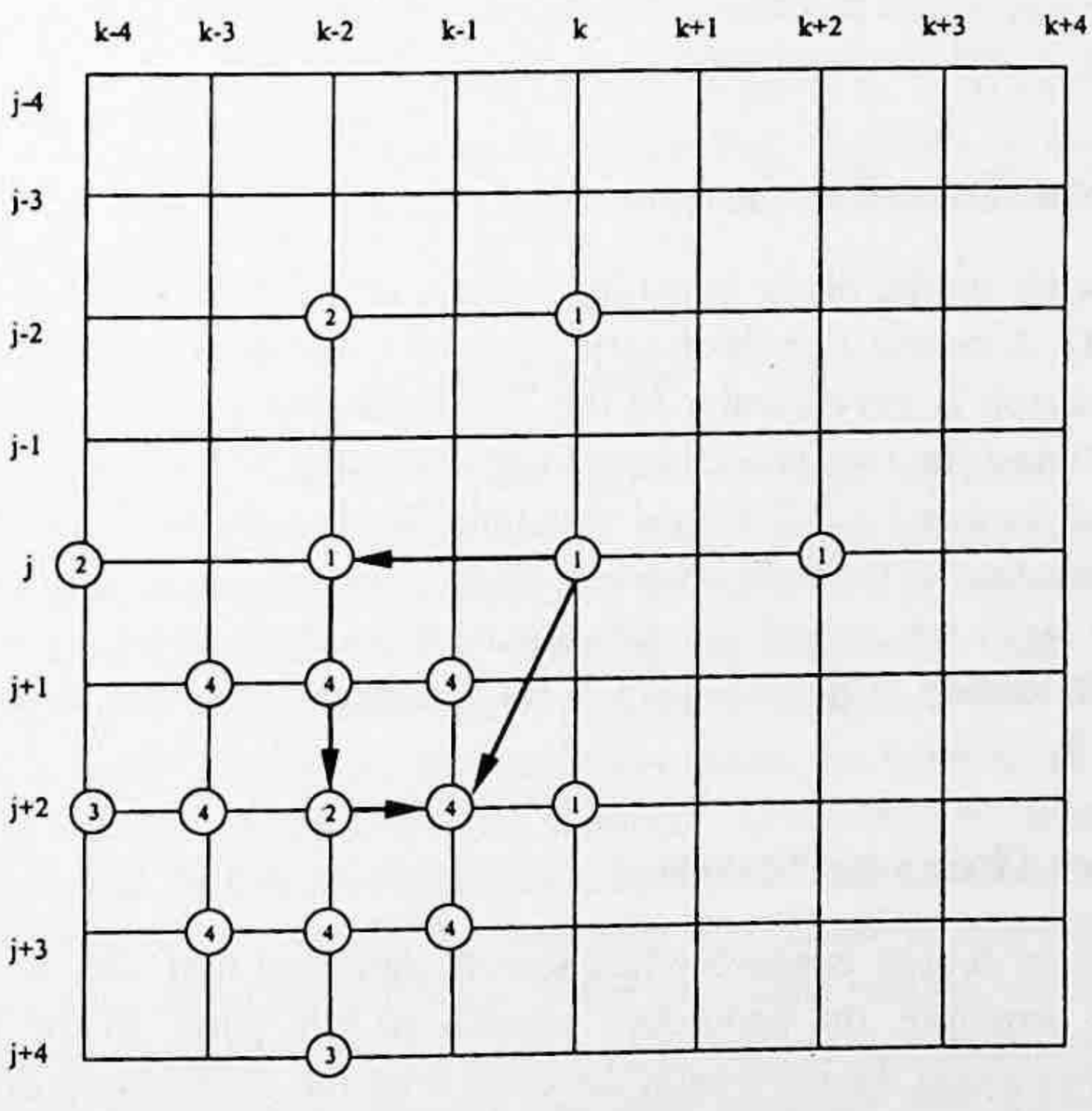
In order to lower computational complexity, several fast searching procedures have been developed. They are introduced below.

11.3.2 2-D Logarithmic Search

Jain and Jain (1981) developed a 2-D logarithmic searching procedure. Based on a 1-D logarithmic search procedure (Knuth, 1973), the 2-D procedure successively reduces the search area, thus reducing the computational burden. The first steps computes the matching criteria for five points in the search window. These five points are as follows: the central point of the search window and the four points surrounding it, with each being a midpoint between the central point and one of the four boundaries of the window. Among these five points, the one corresponding to the minimum dissimilarity is picked as the winner. In the next step, surrounding this winner, another set of five points are selected in a similar fashion to that in the first step, with the distances between the five points remaining unchanged. The exception takes place when either a central point of a set of five points or a boundary point of the search window gives a minimum D value. In these circumstances, the distances between the five points need to be reduced. The procedure continues until the final step, in which a set of candidate points are located in a 3 × 3 2-D grid. Figure 11.3 demonstrates two cases of the procedure. Figure 11.3(a) shows that the minimum D value takes place on a boundary, while Figure 11.3(b) shows the minimum D value in the central position.



(a)



(b)

FIGURE 11.3 (a) A 2-D logarithmic search procedure. Points at (j, k+2), (j+2, k+2), (j+2, k+4), and (j+1, k+4) are found to give the minimum dissimilarity in steps 1, 2, 3, and 4, respectively. (b) A 2-D logarithmic search procedure. Points at (j, k-2), (j+2, k-2), and (j+2, k-1) are found to give the minimum dissimilarity in steps 1, 2, 3, and 4, respectively.

A convergence proof of the procedure is presented by Jain and Jain (1981), under the assumption that the dissimilarity monotonically increases as the search point moves away from the point corresponding to the minimum dissimilarity.

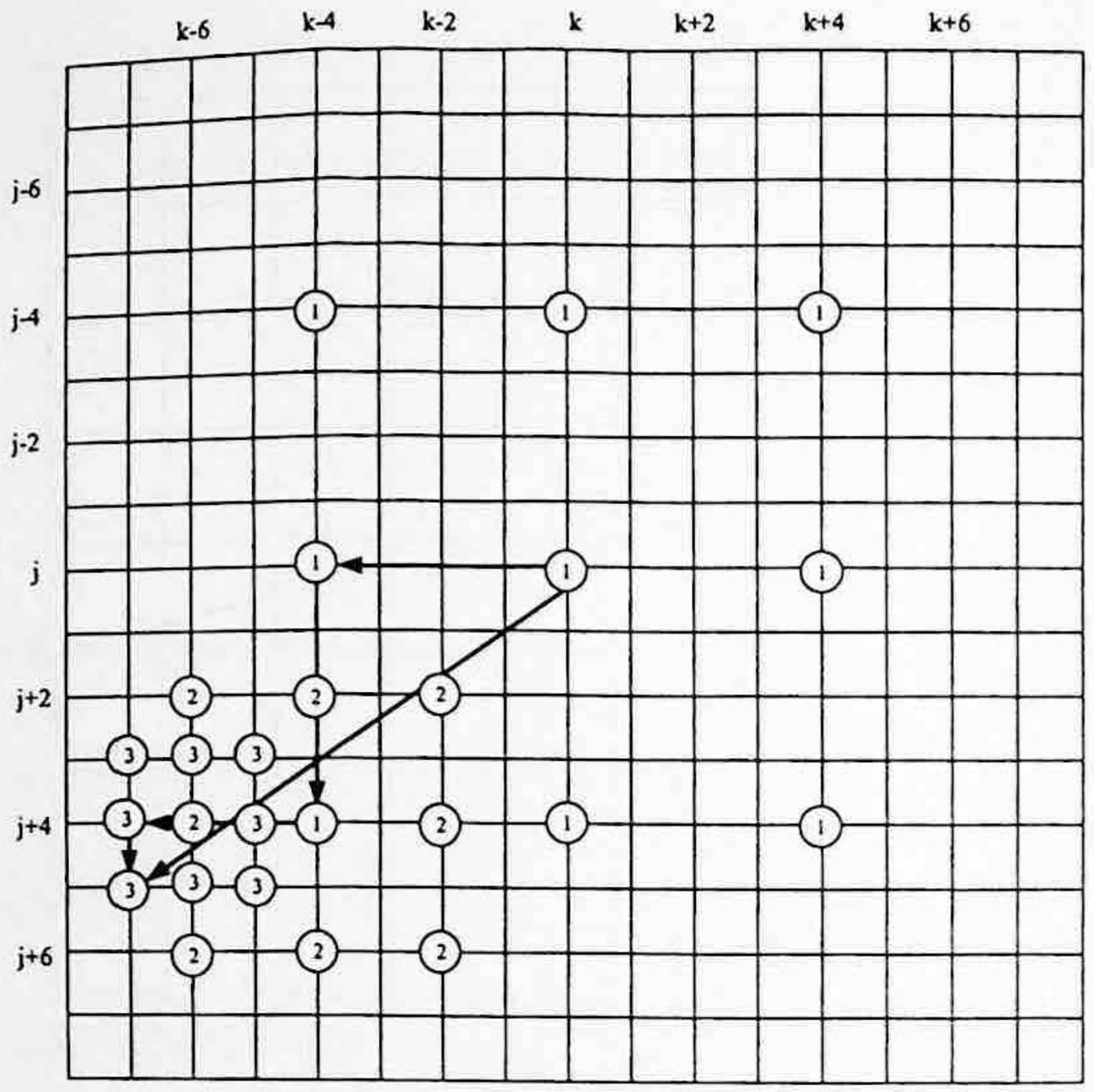


FIGURE 11.4 Three-step search procedure. Points (j+4, k-4), (j+4, k-6), and (j+5, k-7) give the minimum dissimilarity in steps 1, 2, and 3, respectively.

11.3.3 Coarse-Fine Three-Step Search

Another important work on the block matching technique was completed at almost the same time by Koga et al. (1981). A coarse-fine three-step procedure was developed for fast searching.

The three-step search is very similar to the 2-D logarithm search. There are, however, three main differences between the two procedures. First, each step in the three-step search compares a set of nine points that form a 3×3 2-D grid structure. Second, the distances between the points in the 3×3 2-D grid structure in the three-step search decrease monotonically in steps 2 and 3. Third, a total of only three steps are carried out. Obviously, these three items are different from the 2-D logarithmic search described in Section 11.3.2. An illustrative example of the three-step search is shown in Figure 11.4.

11.3.4 Conjugate Direction Search

The conjugate direction search is another fast search algorithm that was developed by Srinivasan and Rao (1984). In principle, the procedure consists of two parts. In the first part, it finds the minimum dissimilarity along the horizontal direction with the vertical coordinate fixed at an initial position. In the second part, it finds the minimum D value along the vertical direction with the horizontal coordinate fixed in the position determined in the first part. Starting with the vertical direction followed by the horizontal direction is, of course, functionally equivalent. It was reported that this search procedure works quite efficiently (Srinivasan and Rao, 1984).

Figure 11.5 illustrates the principle of the conjugate direction search. In this example, each step involves a comparison between three testing points. If a point assumes the minimum D value compared with both of its two immediate neighbors (in one direction), then it is considered to be the best match along this direction, and the search along another direction is started. Specifically, the procedure starts to compare the D values for three points (j, k–1), (j, k), and (j, k+1). If the D value of point (j, k–1) appears to be the minimum among the three, then points (j, k-2), (j, k–1),

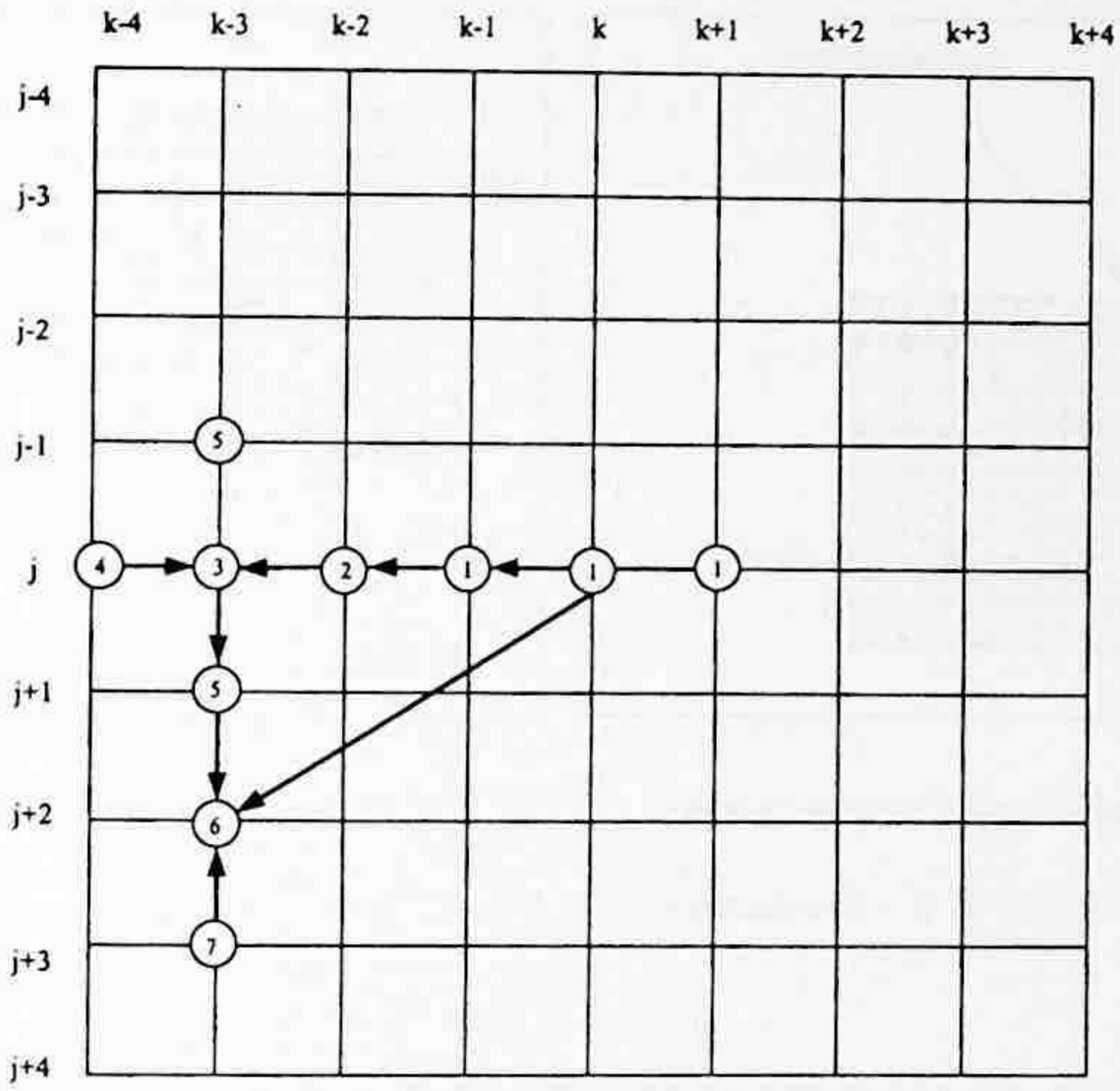


FIGURE 11.5 Conjugate direction search.

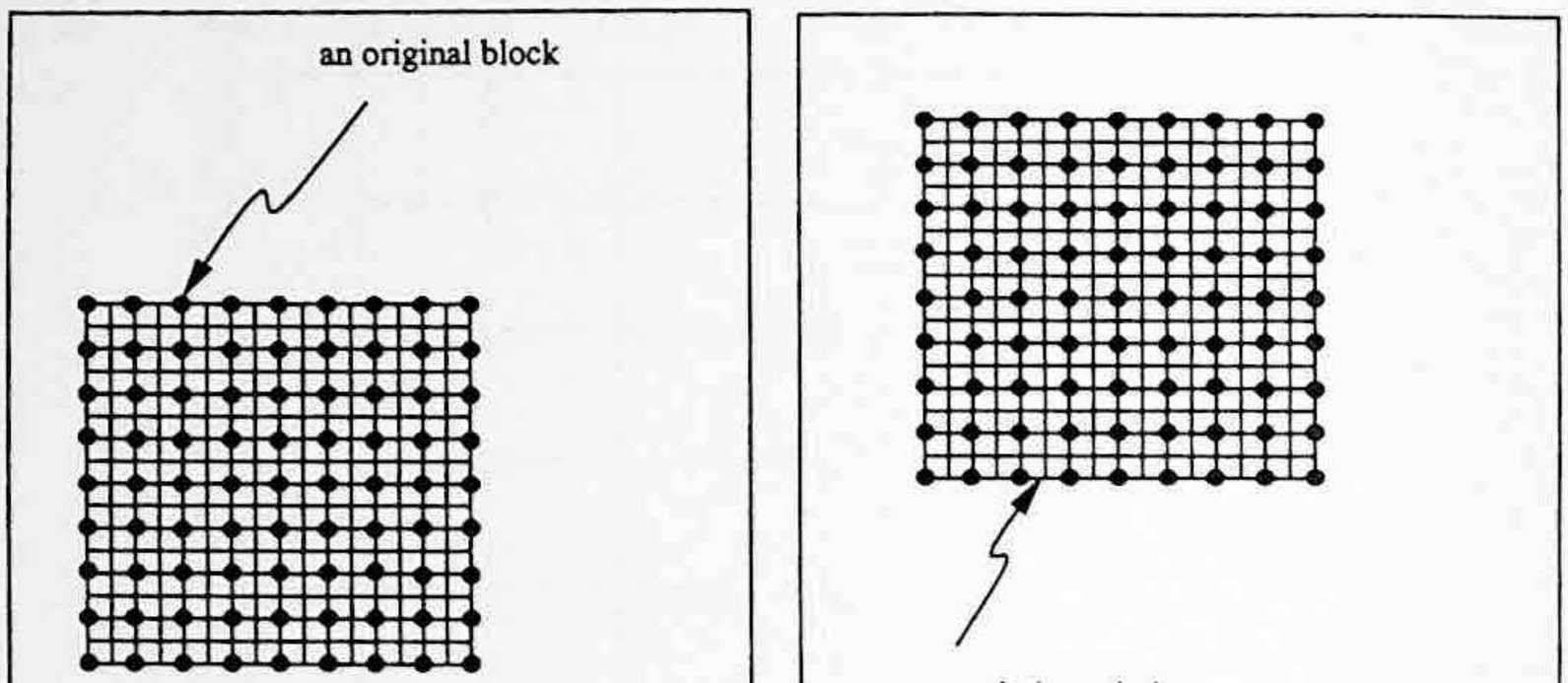
and (j, k) are examined. The procedure continues, finding point (j, k–3) as the best match along the horizontal direction since its D value is smaller than that of points (j, k–4) and (j, k–2). The procedure is then conducted along the vertical direction. In this example the best matching is finally found at point (j+2, k–3).

11.3.5 Subsampling in the Correlation Window

In the evaluation of the matching criterion, either MAD or MSE, all pixels within a correlation window at the t_{n-1} frame and an original block at the t_n frame are involved in the computation. Note that the correlation window and the original block are the same size (refer to Figure 11.1). In order to further reduce the computational effort, a subsampling inside the window and the block is performed (Bierling, 1988). Aliasing effects can be avoided by using low-pass filtering. For instance, only every second pixel, both horizontally and vertically inside the window and the block, is taken into account for the evaluation of the matching criterion. Obviously, by using this subsampling technique, the computational burden is reduced by a factor of 4. Since 3/4 of the pixels within the window and the block are not involved in the matching computation, however, the use of such a subsampling procedure may affect the accuracy of the estimated motion vectors, especially in the case of small-size blocks. Therefore, the subsampling technique is recommended only for those cases with a large enough block size so that the matching accuracy will not be seriously affected. Figure 11.6 shows an example of 2×2 subsampling applied to both an original block of 16×16 at the t_n frame and a correlation window of the same size at the t_{n-1} frame.

11.3.6 Multiresolution Block Matching

It is well known that a multiresolution structure, also known as a pyramid structure, is a very powerful computational configuration for various image processing tasks. To save computation in block matching, it is natural to resort to the pyramid structure. In fact, the multiresolution technique has been regarded as one of the most efficient methods in block matching (Tzovaras et al., 1994). In a named top-down multiresolution technique, a typical Gaussian pyramid is formed first.



(a) An original block of 16×16 in frame at t_n (b) A correlation window of 16×16 in frame at t

FIGURE 11.6 An example of 2×2 subsampling in the original block and correlation window for a fast search.

Before diving into further description, let us pause here to give those readers who have not been exposed to the Gaussian pyramid a short introduction to the concept. For those who know the concept, this paragraph can be skipped. Briefly speaking, a Gaussian pyramid can be understood as a set of images with different resolutions related to an original image in a certain way. The original image has the highest resolution and is considered as the lowest level, sometimes called the bottom level, in the set. From the bottom level to the top level, the resolution decreases monotonically. Specifically, between two consecutive levels, the upper level is half as large as the lower level in both horizontal and vertical directions. The upper level is generated by applying a low-pass filter (which has a group of weights) to the lower level, followed by a 2×2 subsampling. That is, each pixel in the upper level is a weighted average of some pixels in the lower level. In general, this iterative procedure of generating a level in the set is equivalent to convolving a specific weight function with the original image at the bottom level followed by an appropriate subsampling. Under certain conditions, these weight functions can closely approximate the Gaussian probability density function, which is why the pyramid is named after Gauss. (For a detailed discussion, readers are referred to Burt and Adelson [1983, 1984].) A Gaussian pyramid structure is depicted in Figure 11.7. Note that the Gaussian pyramid depicted in Figure 11.7 resembles a so-called quad-tree structure in which each node has four children nodes. In the simplest quad-tree pyramid, each pixel in an upper level is assigned an average value of its corresponding four pixels in the next lower level.

Now let's return to our discussion on the top-down multiresolution technique. After a Gaussian pyramid has been constructed, motion search ranges are allocated among the different pyramid levels. Block matching is initiated at the lowest resolution level to obtain an initial estimation of motion vectors. These computed motion vectors are then propagated to the next higher resolution level, where they are corrected and then propagated to the next level. This procedure continues until the highest resolution level is reached. As a result, a large amount of computation can be saved. Tzovaras et al. (1994) showed that a two-level Gaussian pyramid outperforms a three-level pyramid. Compared with full search block matching, the top-down multiresolution block search saves up to 67% of computations without seriously affecting the quality of the reconstructed images.

In conclusion, it has been demonstrated that multiresolution is indeed an efficient computational structure in block matching. This once again confirms the high computational efficiency of the multiresolution structure.

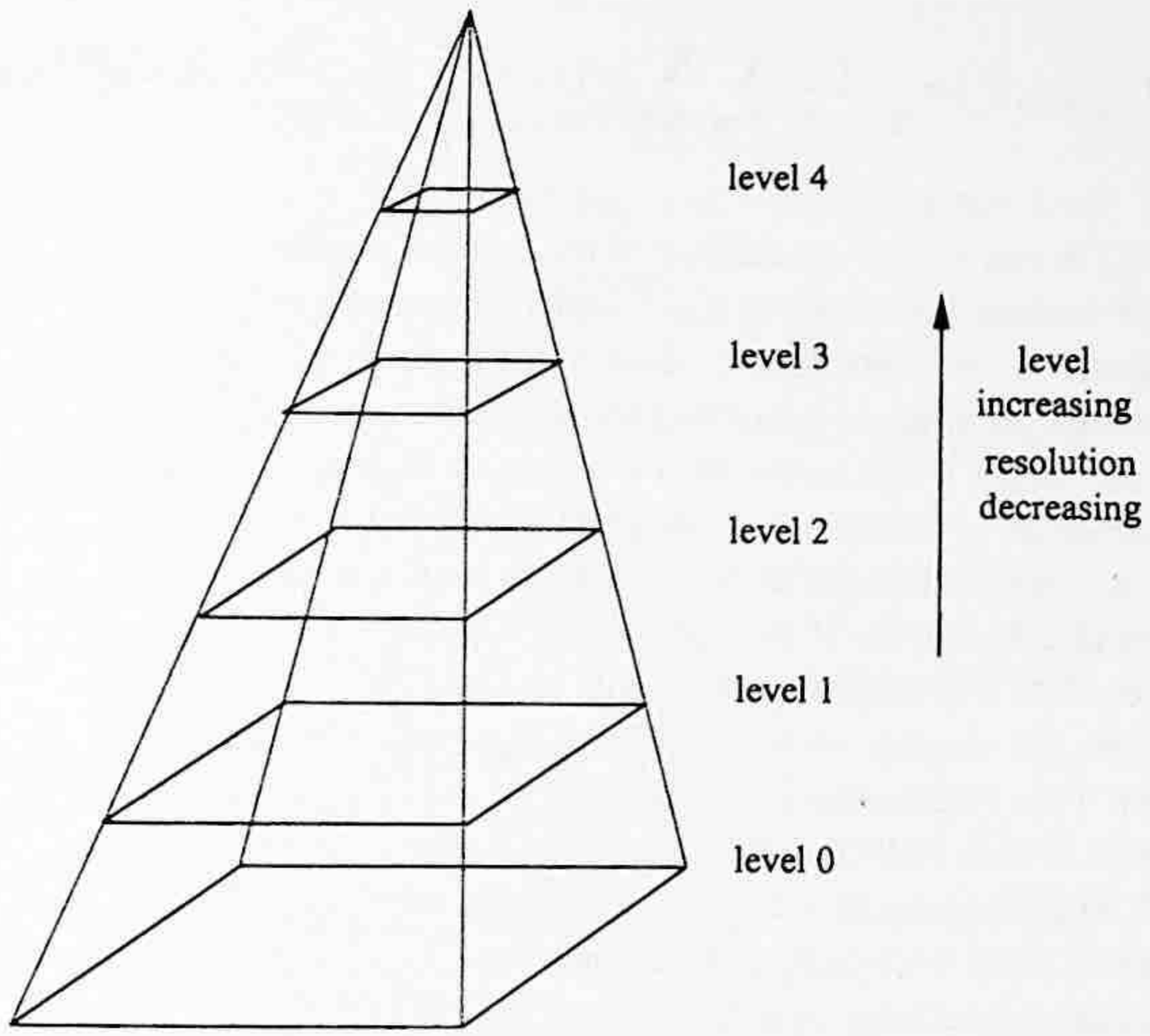


FIGURE 11.7 Gaussian pyramid structure.

11.3.7 Thresholding Multiresolution Block Matching

With the multiresolution technique discussed above, the computed motion vectors at any intermediate pyramid level are projected to the next higher resolution level. In reality, some computed motion vectors at the lower resolution levels may be inaccurate and have to be further refined, while others may be relatively accurate and able to provide satisfactory motion compensation for the corresponding block. From a computation-saving point of view, for the latter class it may not be worth propagating the motion vectors to the next higher resolution level for further processing.

Motivated by the above observation, a new multiresolution block matching method with a thresholding technique was developed by Shi and Xia (1997). The thresholding technique prevents those blocks, whose estimated motion vectors provide satisfactory motion compensation, from further processing, thus saving a lot of computation. In what follows, this technique is presented in detail so as to provide readers with an insight to both multiresolution block matching and thresholding multiresolution block matching techniques.

Algorithm — Let $f_n(x, y)$ be the frame of an image sequence at current moment n. First, two Gaussian pyramids are formed, pyramids n and $n - 1$, from image frames $f_n(x, y)$ and $f_{n-1}(x, y)$, respectively. Let the levels of the pyramids be denoted by l, $l = 0, 1, \ldots, L$, where 0 is the lowest resolution level (top level), L is the full resolution level (bottom level), and $L+1$ is the total number of layers in the pyramids. If (i, j) are the coordinates of the upper-left corner of a block at level l of pyramid n, the block is referred to as block $(i, j)_n^l$. The horizontal and vertical dimensions of a block at level l are denoted by b_x^l and b_y^l, respectively. Like the variable block size method (refer to Method 1 in Tzovaras et al. [1994]), the size of the block in this work varies with the pyramid levels. That is, if the size of a block at level l is b_x^l, then the size of the block at level $l + 1$ becomes $2b_x^l \times 2b_y^l$. The variable block size method is used because it gives more efficient motion estimation than the fixed block size method. Here, the matching criterion used for motion estimation is the MAD because it does not require multiplication and performs similar to the MSE. The MAD between block $(i, j)^l b_n^l$ of the current frame and block $(i + v_x, j + v_y)^l b_{n-l}^l$ of the previous frame at level l can be calculated as

$$MAD_{(i,j)_n^l}\left(v_x^l, v_y^l\right) = \frac{1}{b_x^l \times b_y^l} \sum_{k=0}^{b_x^l-1} \sum_{m=0}^{b_y^l-1} \left| f_n^l(i+k, j+m) - f_{n-1}^l\left(i+k+v_x^l, j+m+v_y^l\right) \right| \qquad (11.5)$$

where $V^l = (v_x^l, v_y^l)$ is one of the candidates of the motion vector of block $(i, j)_n^l$, v_x^l, v_y^l are the two components of the motion vector along the x and y directions, respectively.

A block diagram of the algorithm is shown in Figure 11.8. The threshold in terms of MAD needs to be determined in advance according to the accuracy requirement of the motion estimation. Determining the threshold is discussed below in Part B of this subsection. Gaussian pyramids are formed for two consecutive frames of an image sequence from which motion estimation is desired. Block matching is then performed at the top level with the full-search scheme. The estimated motion vectors are checked to see if they provide satisfactory motion compensation. If the accuracy requirement is met, then the motion vectors will be directly transformed to the bottom level of the pyramid. Otherwise, the motion vectors will be propagated to the next higher resolution level for further refinement. This thresholding process is discussed below in Part C of this subsection. The algorithm continues in this fashion until either the threshold has been satisfied or the bottom level has been reached. The skipping of some intermediate-level calculations provides for computational saving. Experimental work with quite different motion complexities demonstrates that the proposed algorithm reduces the processing time from 14 to 20%, while maintaining almost the same quality in the reconstructed image compared with the fastest existing multiresolution block matching algorithm (Tzovaras et al., 1994).

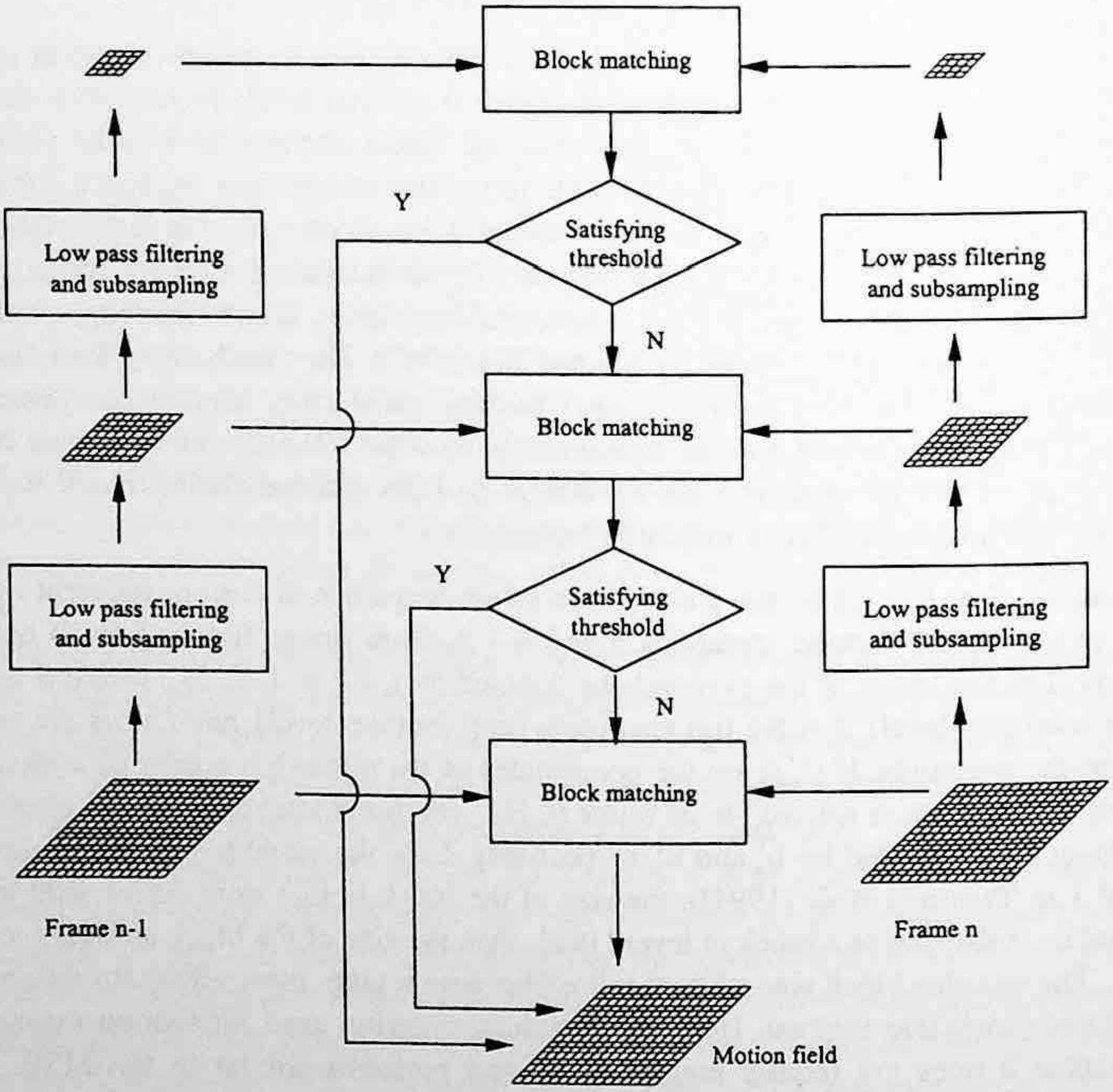


FIGURE 11.8 Block diagram for a three-level threshold multiresolution block matching.

TABLE 11.1
Parameters Used in the Experiments

Parameters at Level	Low Resolution Level	Full Resolution Level
	Miss America	
Search range	3 × 3	1 × 1
Block size	4 × 4	8 × 8
Thresholding value	2	None (not applicable)
	Train	
Search range	4 × 4	1 × 1
Block size	4 × 4	8 × 8
Thresholding value	3	None (not applicable)
	Football	
Search range	4 × 4	1 × 1
Block size	4 × 4	8 × 8
Thresholding value	4	None (not applicable)

Threshold Determination — The MAD accuracy criterion is used in this work for the sake of saving computations. The threshold value has a direct impact on the performance of the proposed algorithm. A small threshold value can improve the reconstructed image quality at the expense of increased computational effort. On the other hand, a large threshold value can reduce the computational complexity, but the quality of the reconstructed image may be degraded. One possible way to determine a threshold value, which was used in many experiments by Shi and Xia (1997), is as follows.

The peak signal-to-noise ratio (PSNR) is commonly used as a measure of the quality of the reconstructed image. As introduced in Chapter 1, it is defined as

$$PSNR = 10 \log_{10} \frac{255^2}{MSE} \tag{11.6}$$

From the given required PSNR, one can find the necessary MSE value. A square root of this MSE value can be chosen as a threshold value, which is applied to the first two images from the sequence. If the resulting PSNR and required processing time are satisfactory, it is then used for the rest of the sequence. Otherwise, the threshold can be slightly adjusted accordingly and applied to the second and third images to check the PSNR and processing time. It was reported in numerous experiments that this adjusted threshold value was accurate enough, and that there was no need for further adjustment. As shown in Table 11.1, the threshold values used for the "Miss America," "Train," and "Football" sequences (three sequences having quite different motion complexities) are 2, 3, and 4, respectively. They are all determined in this fashion and give satisfactory performance, as shown in the three rows marked "New Method (TH=2)," "New Method (TH=3)" and "New Method (TH=4)," respectively, in Table 11.2. That is, the PSNR experiences only about 0.1 dB loss and the processing time decreases drastically. In the experiments, the threshold value of 3, i.e., the average value of 2, 3, and 4, was also tried. Refer to the three rows marked "New Method (TH=3)" in Table 11.2. It is noted that this average threshold value 3 has already given satisfactory performance for all three sequences. Specifically, for the "Miss America" sequence, since the criterion increases from 2 to 3, the PSNR loss increases from 0.12 to 0.48 dB, and the reduction in processing time increases from 20 to 38%. For the "Football" sequence, since the criterion decreases from 4 to 3, the PSNR loss decreases from 0.08 to 0.05 dB, and the reduction in processing time decreases

from 14 to 9%. Obviously, for the "Train" sequence, the criterion, as well as the performance, remains the same. One can therefore conclude that the threshold determination may not require much computation at all.

Thresholding — Motion vectors estimated at each pyramid level will be checked to see if they provide satisfactory motion compensation. Assume $V^l(i, j) = (v_x^l, v_y^l)$ is the estimated motion vector for block $(i, j)^l_n$ at level l of pyramid n. For thresholding, $V^l(i, j)$ should be directly projected to the bottom level L. The corresponding motion vector for the same block at the bottom level of pyramid n will be $V^L(2^{(L-l)} i, 2^{(L-l)} j)$, and is given as

$$V^L\left(2^{(L-l)} i,\ 2^{(L-l)} j\right) = 2^{(L-l)} V^l(i, j) \tag{11.7}$$

The MAD between the block at the bottom pyramid level of the current frame and its counterpart in the previous frame can be determined according to Equation 11.5, where the motion vector is $V^L = V^L(2^{(L-l)} i, 2^{(L-l)} j)$. This computed MAD value can be compared with the predefined threshold. If this MAD value is less than the threshold, the computed motion vector $V^L(2^{(L-l)} i, 2^{(L-l)} j)$ will be assigned to block $(2^{(L-l)}\, i, 2^{(L-l)}\, j)^L_n$ at level L in the current frame and motion estimation for this block will be stopped. If not, the estimated motion vector $V^l(i, j)$ at level l will be propagated to level $l + 1$ for further refinement. Figure 11.9 gives an illustration of the above thresholding process.

Experiments — To verify the effectiveness of the proposed algorithm, extensive experiments have been conducted. The performance of the new algorithm is evaluated and compared with that of Method 1, one of the most efficient multiresolution block matching methods (Tzovaras et al., 1994) in terms of PSNR, error image entropy, motion vector entropy, the number of blocks stopped at the top level vs. the total number of blocks, and processing time. The number of blocks stopped at the top level is the number of blocks withheld from further processing, while the total number of blocks is the number of blocks existing at the top level. It is noted that the total number of blocks is the same for each level in the pyramid. The processing time is the sum of the total number of additions involved in the evaluation of the MAD and the thresholding operation.

In the experiments, two-level pyramids are used since they give better performance for motion estimation purposes (Tzovaras et al., 1994). The algorithms are tested on three video sequences with different motion complexities, i.e., the "Miss America," "Train," and "Football." The "Miss America" sequence has a speaker imposed on a static background and contains less motion. The "Train" sequence has more detail and contains a fast-moving object (train). The 20th frame of the sequence is shown in Figure 11.10. The "Football" sequence contains the most complicated motion

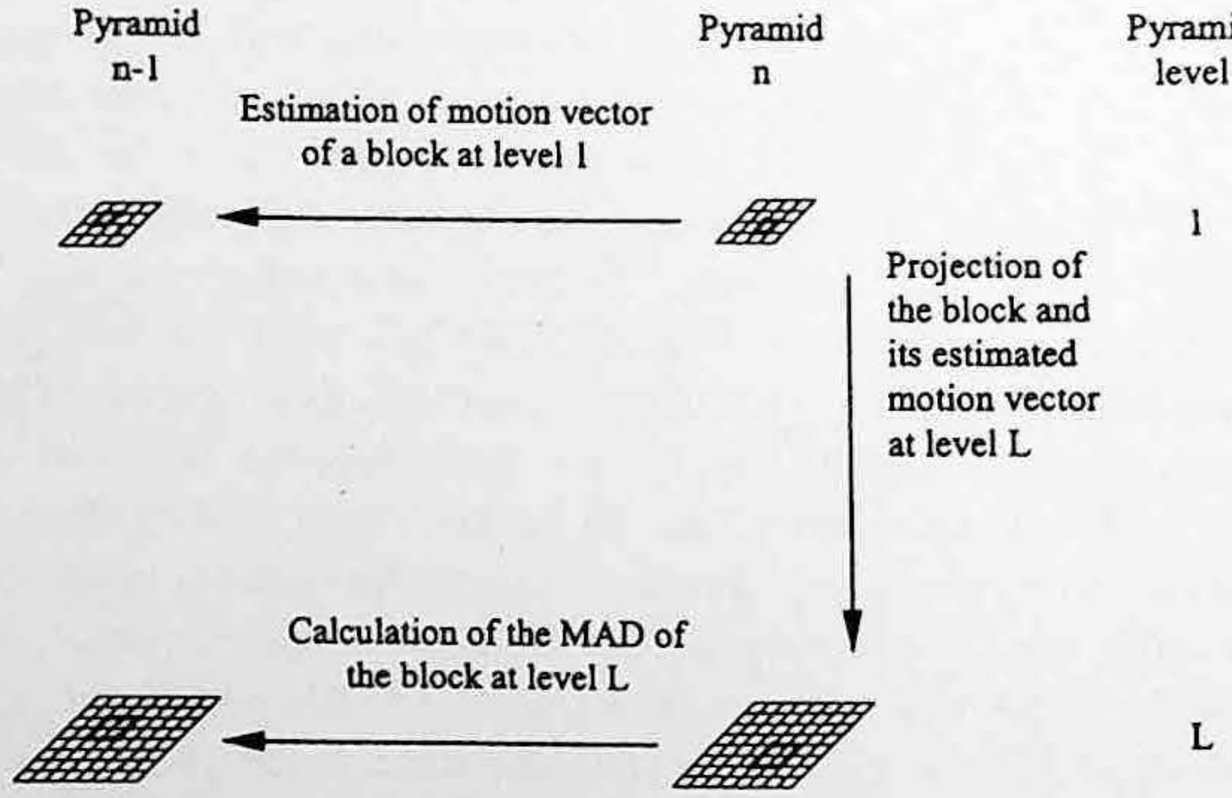


FIGURE 11.9 The thresholding process.

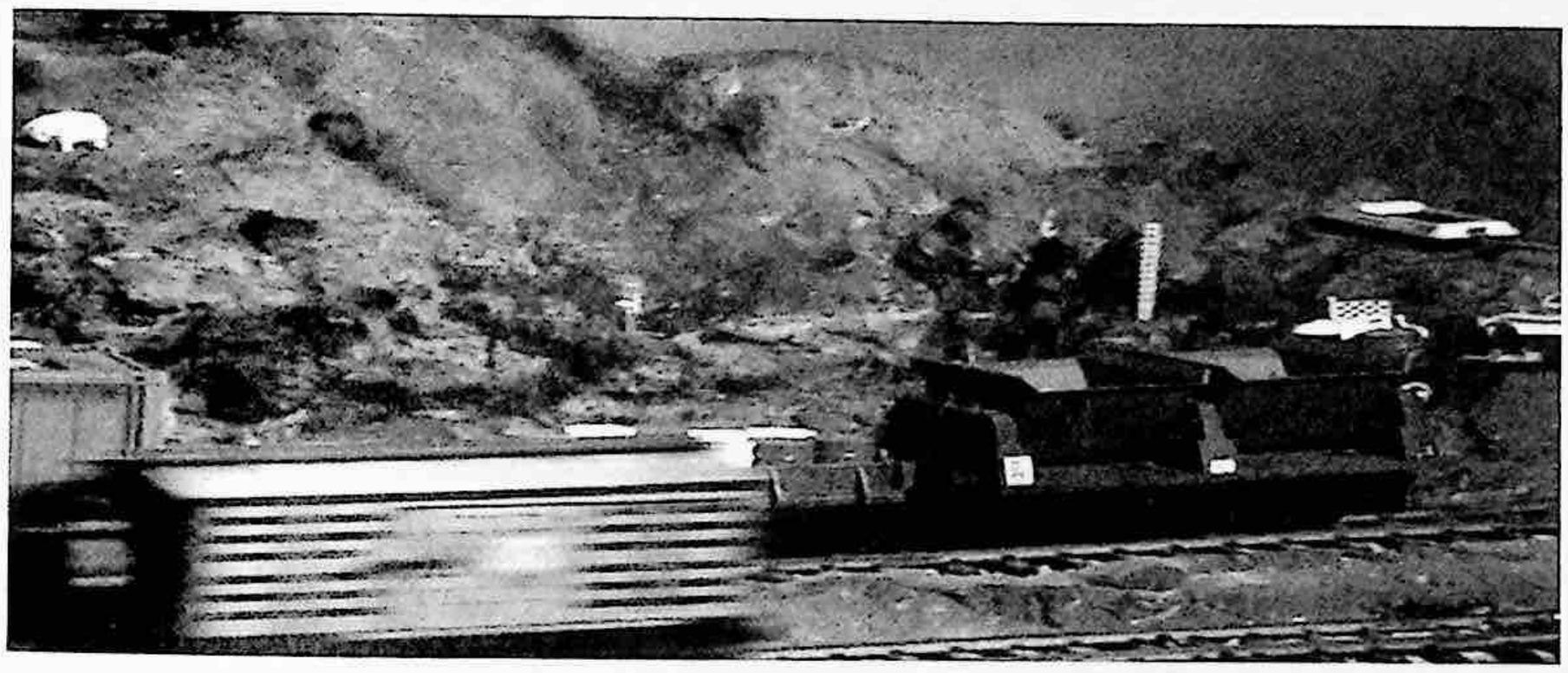


FIGURE 11.10 The 20th frame of the "Train" sequence.

FIGURE 11.11 The 20th frame in the "Football" sequence.

compared with the other two sequences. The 20th frame is shown in Figure 11.11. Table 11.1 is the list of implementing parameters used in the experiments. Tables 11.2 and 11.3 give the performance of the proposed algorithm compared with Method 1. In all three cases, the motion estimation has a half-pixel accuracy, the meaning of which will be explained in the next section. All performance measures listed there are averaged for the first 25 frames of the testing sequences.

Each frame of the "Miss America" sequence is of 360×288 pixels. For convenience, only the central portion, 320×256 pixels, is processed. Using the operational parameters listed in Table 11.1 (with a criterion value of 2), 38% of the total blocks at the top level satisfy the predefined criterion and are not propagated to the bottom level. The processing time needed by the proposed algorithm is 20% less than Method 1, while the PSNR, the error image entropy, and the vector entropy are almost the same. Compared with Method 1, an extra amount of computation (around 0.16×10^6

TABLE 11.2
Experimental Results (I)

	PSNR (dB)	Error Image Entropy (bits per pixel)	Vector Entropy (bits/vector)	Block Stopped at Top Level/Total Block	Processing Times (No. of Additions, 10^6)
Miss America Sequence					
Method 1 (Tzovaras et al., 1994)	38.91	3.311	6.02	0/1280	10.02
New method (TH=2)	38.79	3.319	5.65	487/1280	8.02
New method (TH=3)	38.43	3.340	5.45	679/1280	6.17
Train Sequence					
Method 1 (Tzovaras et al., 1994)	27.37	4.692	6.04	0/2560	22.58
New method (TH=3)	27.27	4.788	5.65	1333/2560	18.68
Football Sequence					
Method 1 (Tzovaras et al., 1994)	24.26	5.379	7.68	0/3840	30.06
New method (TH=4)	24.18	5.483	7.58	1464/3840	25.90
New method (TH=3)	24.21	5.483	7.57	1128/3840	27.10

additions) is conducted on the thresholding operation, but a large computational savings (around 2.16×10^6 additions) is achieved by withholding from further processing those blocks whose MAD values at the full resolution level are less than the predefined accuracy criterion.

The frames of the "Train" sequence are 720×288 pixels, and only the central portion, 640×256 pixels, is processed. Using the operational parameters listed in Table 11.1 (with a criterion value of 3), about 52% of the total blocks are stopped at the top level. The processing time is reduced about 17% by the new algorithm, compared with Method 1. The PSNR, the error image entropy, and the vector entropy are almost the same.

The frames of the "Football" sequence are 720×480 pixels, and only the central portion, 640×384 pixels, is processed. Using the operational parameters listed in Table 11.1 (with a criterion value of 4), about 38% of the total blocks are stopped at the top level. The processing time is about 14% less than that required by Method 1, while the PSNR, the error image entropy, and the vector entropy are almost the same.

As discussed, the experiments with a single accuracy criterion of 3 also produce similarly good performance for the three different image sequences.

In summary, it is clear that with the three different testing sequences, the thresholding multiresolution block matching algorithm works faster than the fastest existing top-down multiresolution block matching algorithm while achieving almost the same quality of the reconstructed image.

11.4 MATCHING ACCURACY

Apparently, the two components of the displacement vectors obtained using the technique described above are an integer multiple of pixels. This is referred to as one-pixel accuracy. If a higher accuracy is desired, i.e., the components of the displacement vectors may be a non-integer multiple of pixels, then spatial interpolation is required. Not only will more computation be involved, but also more bits will be required to represent motion vectors. The gain is a more accurate motion estimation, hence less prediction error. In practice, half-pixel or quarter-pixel accuracy are two widely utilized accuracies other than one-pixel accuracy.

11.5 LIMITATIONS WITH BLOCK MATCHING TECHNIQUES

Although very simple, straightforward, and efficient, hence, utilized most widely in video coding, the block matching motion compensation technique has its drawbacks. First, it has an unreliable motion vector field with respect to the true motion in 3-D world space. In particular, it has unsatisfactory motion estimation and compensation along moving boundaries. Second, it causes block artifacts. Third, it needs to handle side information. That is, it needs to encode and transmit motion vectors as an overhead to the receiving end, thus making it difficult to use smaller block size to achieve higher accuracy in motion estimation.

All these drawbacks are due to its simple model: each block is assumed to experience a uniform translation and the motion vectors of partitioned blocks are estimated independently of each other. Unreliable motion estimation, particularly along moving boundaries, causes more prediction error, hence reduced coding efficiency.

The block artifacts do not cause severe perceptual degradation to the human visual system (HVS) when the available coding bit rate is adequately high. This is because, with a high bit rate, a sufficient amount of the motion-compensated prediction error can be transmitted to the receiving end, hence improving the subjective visual effect to such an extent that the block artifacts do not appear to be annoying. However, when the available bit rate is low, particularly lower than 64 kbps, the artifacts become visually unpleasant. In Figure 11.12, a reconstructed frame of the "Miss America" sequence at a low bit rate is shown. Obviously, block artifacts are very annoying,

FIGURE 11.12 The 21st reconstructed frame of the "Miss America" sequence using a codec following H.263.

especially where the mouth and hair are involved. The sequence was coded and decoded by using a codec following ITU-T Recommendations H.263, an international standard in which block matching is utilized for motion estimation.

The assumption that motion within each block is uniform requires a small block size such as 16×16 and 8×8. A small block size leads to a large number of motion vectors, however, resulting in a large overhead of side information. A study by Chan et al. (1990) indicated that 8×8 block matching performs much better than 16×16 in terms of decoded image quality due to better motion estimation and compensation. The bits used for encoding motion vectors, however, increase significantly (about four times), which may be prohibitive for very low bit rate coding since the total bit rate needed for both prediction error and motion vectors may exceed the available bit rate. It is noted that when the coding bit rate is quite low, say, on the order of 20 kbps, the *side* information becomes compatible with the *main* information (prediction error) (Lin et al., 1997).

Tremendous research efforts have been made to overcome the limitations of block-matching techniques. Some improvements have been achieved and are discussed next. It should be kept in mind, however, that block matching is still by far the most popular and efficient motion estimation and compensation technique utilized for video coding, and it has been adopted for use by various international coding standards. In other words, block matching is the most appropriate technique in the framework of first-generation video coding (Dufaux and Moscheni, 1995).

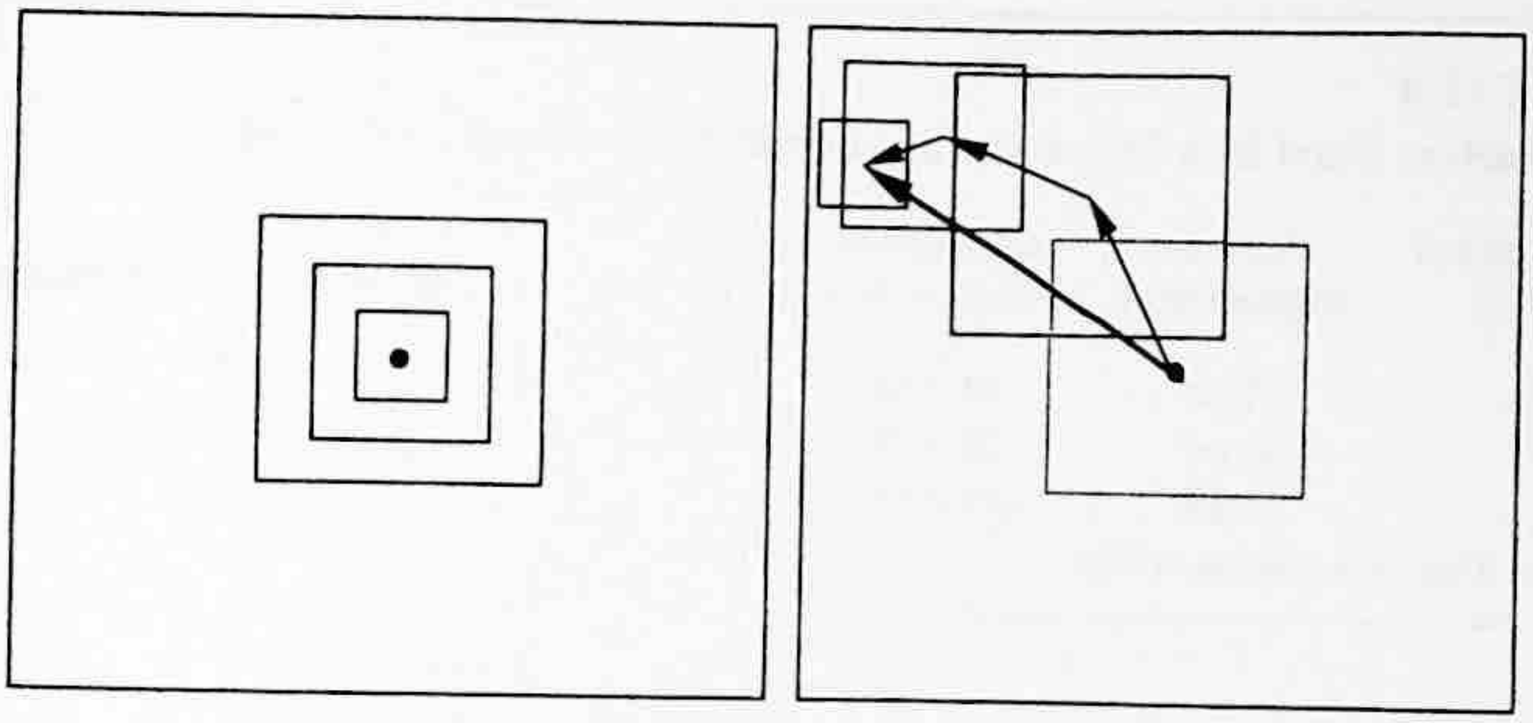
11.6 NEW IMPROVEMENTS

11.6.1 Hierarchical Block Matching

Bierling (1988) developed the hierarchical search based on the following two observations. On the one hand, for a relatively large displacement, accurate block matching requires a relatively large block size. This is conceivable if one considers its opposite case: a large displacement with a small correlation window. Under this circumstance, the search range is large. Therefore the probability of finding multiple matches is high, resulting in unreliable motion estimation. On the other hand, a large block size may violate the assumption that all pixels in the block share the same displacement vector. Hence a relatively small block size is required in order to meet the assumption. These observations shed light on the problem of using a fixed block size, which may lead to unreliable motion estimation.

To satisfy these two contradicting requirements simultaneously, in a hierarchical search procedure a set of different sizes of blocks and correlation windows is utilized. To facilitate the discussion, consider a three-level hierarchical block-matching algorithm, in which three block-matching procedures are conducted, each with its own parameters. Block matching is first conducted with respect to the largest size of blocks and correlation windows. Using the estimated displacement vector as an initial vector at the second level, a new search is carried out with respect to the second largest size of blocks and correlation windows. The third search procedure is carried out similarly, based on the results of the second search. An example with three correlation windows is illustrated in Figure 11.13. It is noted that the resultant displacement vector is the sum of the three displacement vectors determined by three searches.

The parameters in these three levels are listed in Table 11.4. The algorithm is described below with an explanation of the various parameters in Table 11.4. Prior to each block matching, a separate low-pass filter is applied to the whole image in order to achieve reliable block matching. The low-pass filtering used is simply a local averaging. That is, the gray value of every pixel is replaced by the mean value of the gray values of all pixels within a square area centered at the pixel to which the mean value is assigned. In calculating the matching criterion D value, a subsampling is applied to the original block and the correlation window in order to save computation, which was discussed in Section 11.3.5.



(a) frame t_i (b) frame t_{i-1}

FIGURE 11.13 Hierarchical block matching.

TABLE 11.3
Experimental Results (II)

Frames Tested	Total Blocks Stopped at Top Level (%)	Saved Processing Time Compared with Method 1 in Tzovaras et al. (1994) (%)
"Miss America" sequence (TH = 2)	38	20
"Train" sequence (TH = 3)	52	17
"Football" sequence (TH = 4)	38	14

In the first level, for every 8th pixel horizontally and vertically (a step size of 8×8), block matching is conducted with the maximum displacement being ± 7 pixels, a correlation window size of 64×64, and a subsampling factor of 4×4. A 5×5 averaging low-pass filter is applied prior to first level block matching. Second-level block matching is conducted with respect to every 4th pixel horizontally and vertically (a step size of 4×4). Note that for a pixel whose displacement vector estimate has not been determined in first-level block matching, an average of the four nearest neighboring estimates will be taken as its estimate. All the parameters for the second level are listed in Table 11.4. One thing that needs to be emphasized is that in block matching at this level the search window should be displaced by the estimated displacement vector obtained in the first level. Third-level block matching is dealt with accordingly for every 2nd pixel horizontally and vertically (a step size of 2×2). The different parameters are listed in Table 11.4. In each of the three levels, the three-step search discussed in Section 11.3.3 is utilized.

Experimental work has demonstrated a more reliable motion estimation due to the usage of a set of different sizes for both the original block and the correlation window. The first level with a large window size and a large displacement range determines a major portion of the displacement vector reliably. The successive levels with smaller window sizes and smaller displacement ranges are capable of adaptively estimating motion vectors more locally.

Figure 11.14 shows a portion of an image with pixels processed in the three levels, respectively. It is noted that it is possible to apply one more interpolation after these three levels so that a motion vector field of full resolution is available. Such a full-resolution motion vector field is useful in

TABLE 11.4
Parameters Used in a Three-Level Hierarchical Block Matching

Hierarchical Level	Maximum Displacement	Correlation Window Size	Step Size	LPF Window Size	Subsampling
1	±7 pel	64 × 64	8	5 × 5	4 × 4
2	±3 pel	28 × 28	4	5 × 5	4 × 4
3	±1 pel	12 × 12	2	3 × 3	2 × 2

Source: Data from Bierling (1988).

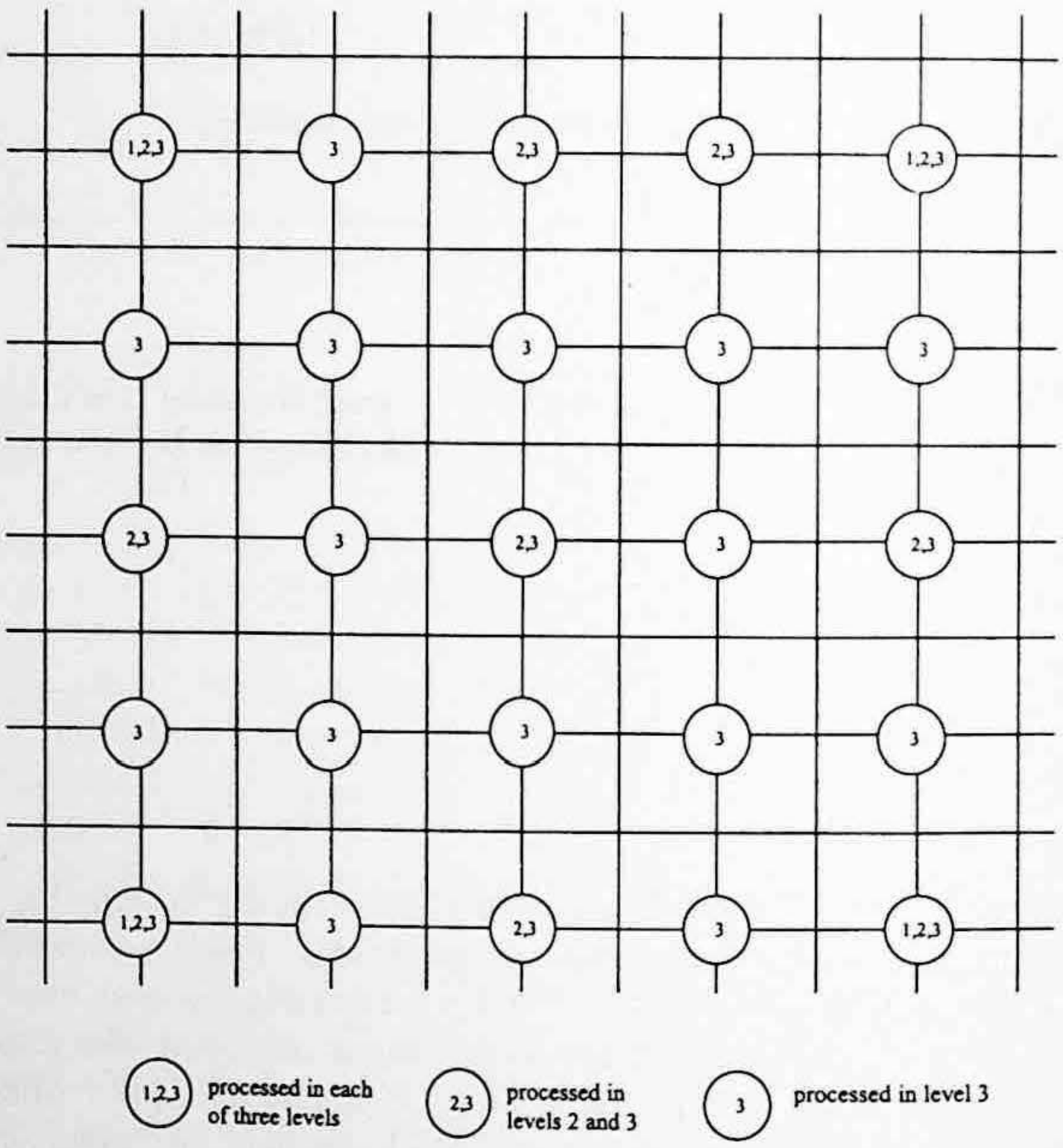


FIGURE 11.14 A portion of an image with pixels processed in all three levels.

such applications as motion-compensated interpolation in the context of videophony. There, in order to maintain a low bit rate some frames are skipped for transmission. At the receiving end these skipped frames need to be interpolated. As discussed in Chapter 10, motion-compensated interpolation is able to produce better frame quality than that achievable by using weighted linear interpolation.

11.6.2 Multigrid Block Matching

Multigrid theory was developed originally in mathematics (Hackbusch and Trottenberg, 1982). It is a useful computational structure in image processing besides the multiresolution one described in Section 11.3.6. A diagram with three different levels used to illustrate a multigrid structure is shown in Figure 11.15. Although it is also a hierarchical structure, each level within the hierarchy

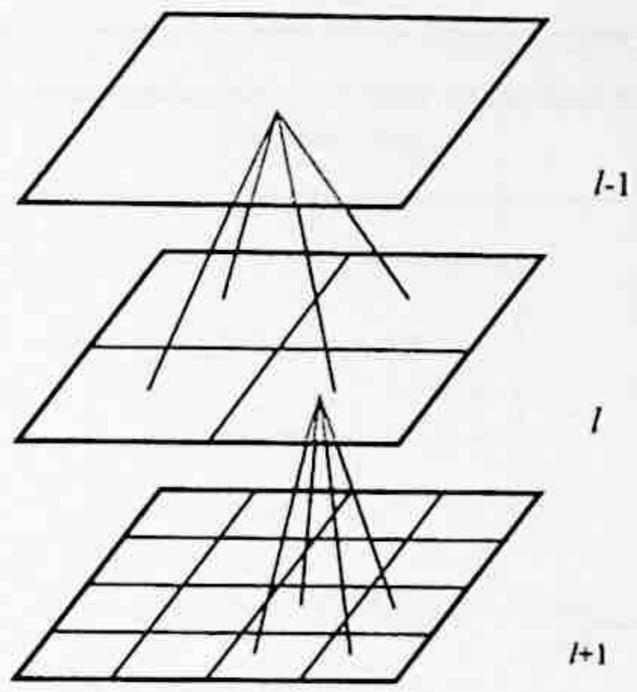


FIGURE 11.15 Illustration of a three-level hierarchical structure.

is of the same resolution. A few algorithms based on multigrid structure have been developed in order to improve the block-matching technique. Two advanced methods are introduced below.

Thresholding Multigrid Block Matching — Realizing that the simple block-based motion model (assuming a uniform motion within a fixed-size block) in the block matching technique causes several drawbacks, Chan et al. (1990) proposed a variable size block matching technique. The main idea is using a split-and-merge strategy with a multigrid structure in order to segment an image into a set of variable size blocks, each of which has an approximately uniform motion. A binary tree (also known as bin-tree) structure is used to record the relationship between these blocks of different sizes.

Specifically, an image frame is initially split into a set of square blocks by cutting the image alternately horizontally and vertically. With respect to each block thus generated, a block matching is performed in conjunction with its previous frame. Then the matching accuracy in terms of the sum squared error is compared with a preset threshold. If it is smaller than or equal to the threshold, the block remains unchanged in the whole process and the estimated motion vector is final. Otherwise, the block will be split into two blocks, and a new run of block matching is conducted for each of these two children blocks. The process continues until either the estimated vector satisfies a preset accuracy requirement or the block size has reached a predefined minimum. At this point, a merge process is proposed by Chan et al. Neighboring blocks under the same intermediate nodes in the bin-tree are checked to see if they can be merged, i.e., if the merged block can be approximated with adequate accuracy by a block in the reconstructed previous frame. It is noted that the merge operation may be optional depending on the specific application.

A block diagram of multigrid block matching is shown in Figure 11.16. Note that it is similar to that shown in Figure 11.8 for the thresholding multiresolution block matching discussed in Section 11.3.6. This observation reflects the similarities between multigrid and multiresolution structures: both are hierarchical in nature and the splitting and merging can be easily performed. An example of an image decomposition and its corresponding bin-tree are shown in Figure 11.17.

It was reported by Chan et al. (1990) that, with respect to a picture of a computer mouse and a coin, the proposed variable size block matching achieves up to a 6-dB improvement in SNR and about 30% reduction in required bits compared with fixed-size (16×16) block matching. For several typical videoconferencing sequences, the proposed algorithm constantly performs better than the fixed-size block matching technique in terms of improved SNR of reconstructed frames with the same bit rate.

A similar algorithm was reported by Xia and Shi (1996) where a quad-tree-based segmentation is used. The thresholding technique is similar to that used by Shi and Xia (1997) and the emphasis is placed on the reduction of computational complexity. It was found that for the head-shoulder type of videophony sequences the thresholding multigrid block matching algorithm performs better

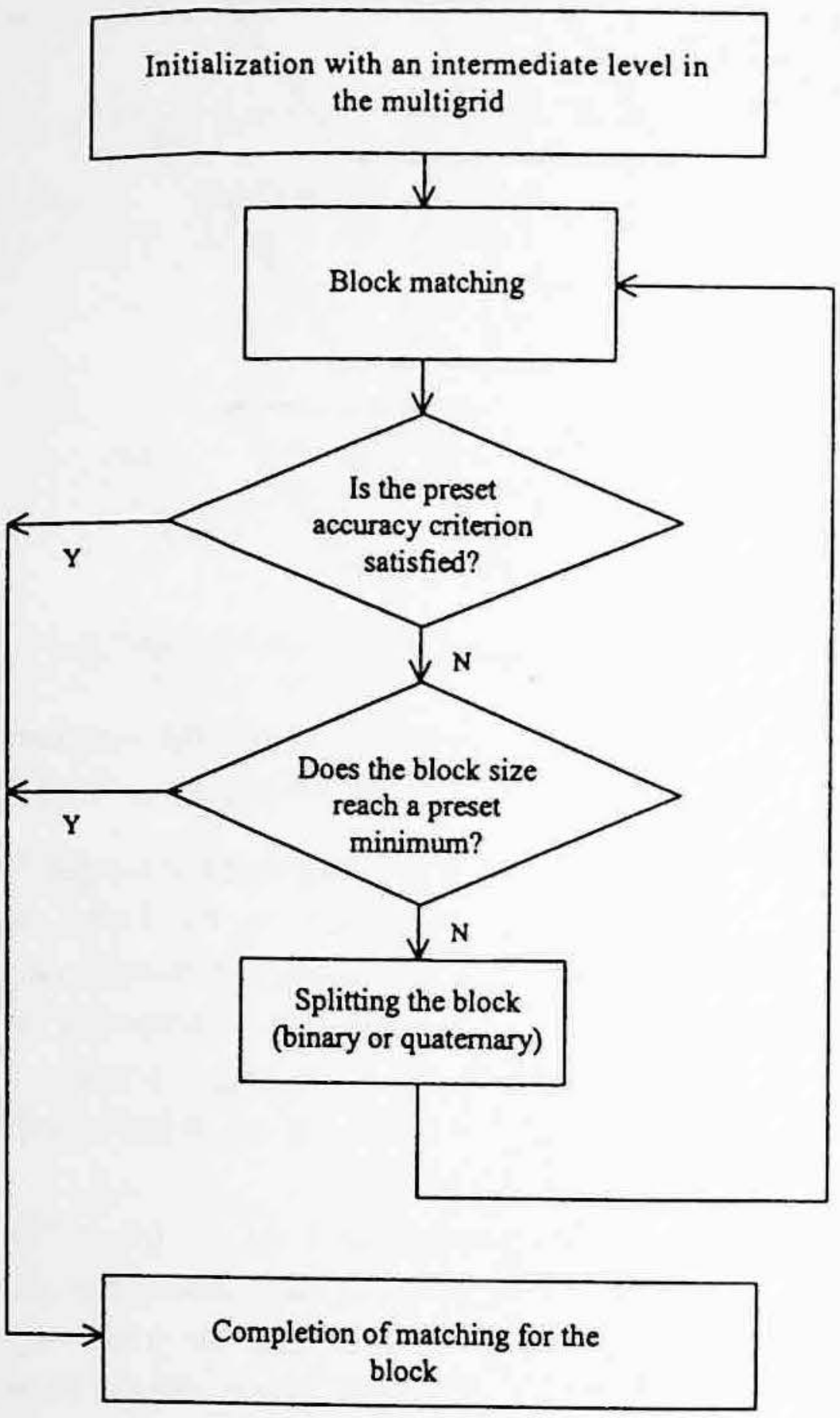


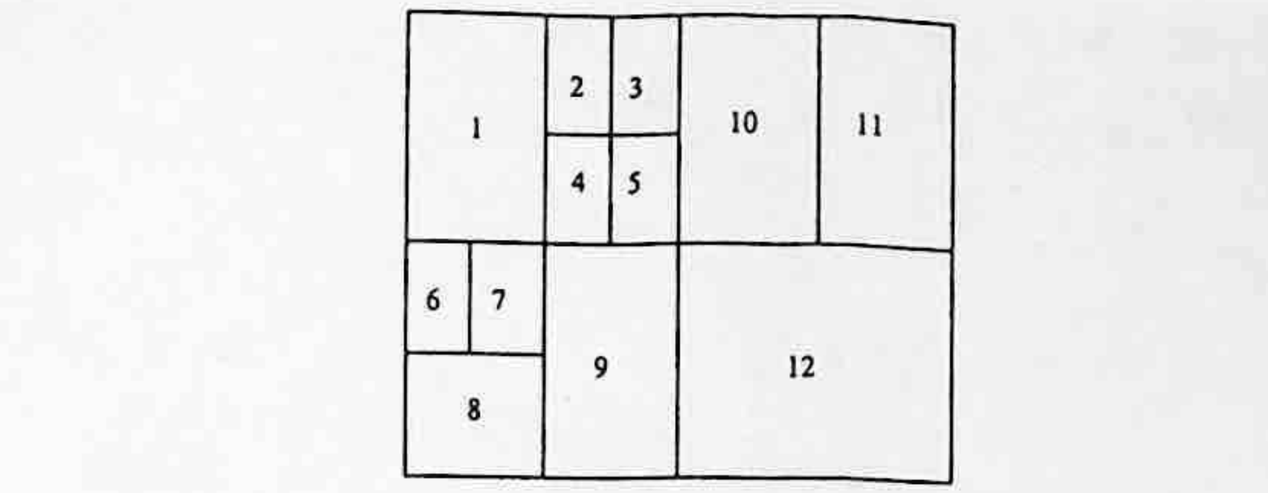
FIGURE 11.16 Flow chart of multigrid block matching.

than the thresholding multiresolution block matching algorithm. For video sequences that contain more complicated details and motion, however, the performance comparison turns out to be reversed.

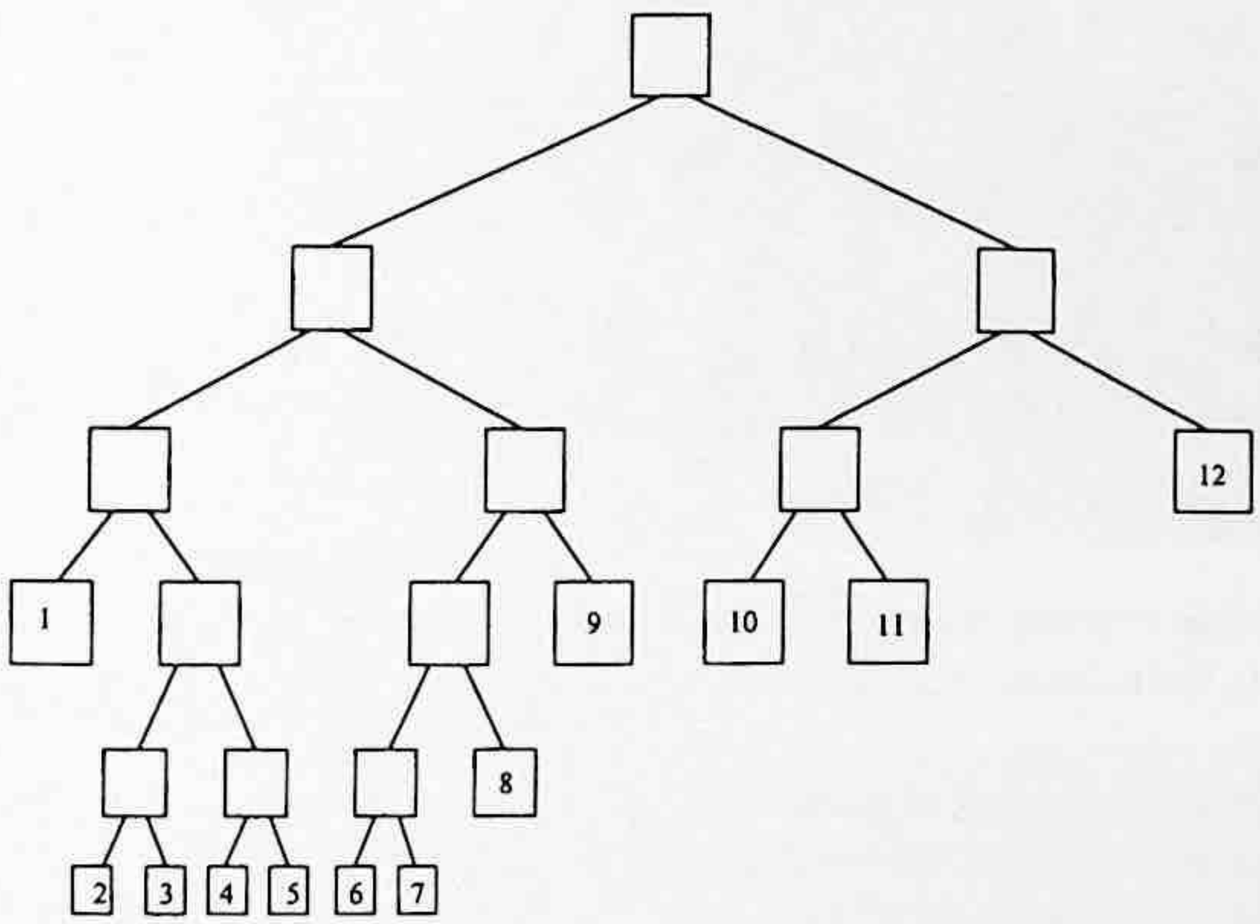
A few remarks can be made as a conclusion for the thresholding technique. Although it needs to encode and transmit the bin-tree or quad-tree as a portion of side information, and it has to resolve the preset threshold issue, overall, the proposed algorithms achieve better performance compared with fixed-size block matching. With the flexibility provided through the variable-size methodology, the proposed approach is capable of making the model of the uniform motion within each block more accurate than fixed-size block matching can do.

Optimal Multigrid Block Matching — As pointed out in Chapter 10, the ultimate goal of motion estimation and motion compensation in the context of video coding is to provide a high code efficiency in real time. In other words, accurate true motion estimation is not the final goal, although accurate motion estimation is certainly desired. This point was presented by Bierling (1988) as well. There, the different requirements with respect to motion-compensated coding and motion-compensated interpolation were discussed. While the former requires motion vector estimation leading to minimum prediction error and at the same time a low amount of motion vector information, the latter requires accurate estimation of true vectors and a high resolution of the motion vector field.

This point was very much emphasized by Dufaux and Moscheni (1995). They clearly stated that in the context of video coding, estimation of true motion in 3-D world space is not the ultimate



(a) An example of a decomposition



(b) The corresponding bin-tree

FIGURE 11.17 Thresholding multigrid block matching.

goal. Instead, motion estimation should be able to provide good temporal prediction and at the same time require low overhead information. In a word, the total amount of information that needs to be encoded should be minimized. Based on this observation, a multigrid block matching technique with an advanced entropy criterion was proposed.

Since it belongs to the category of thresholding multigrid block matching, it shares many similarities with those of Chan et al. (1990) and Xia and Shi (1996). It also bears some resemblance to thresholding multiresolution block matching (Shi and Xia, 1997). What really distinguishes this approach from other algorithms is its segmentation decision rule. Instead of a preset threshold, the algorithm works with an adaptive entropy criterion, which aims at controlling the segmentation in order to achieve an optimal solution in such a way that the total number of bits needed for representing both the prediction error and motion overhead is minimized. The decision of splitting a block is made only when the extra motion overhead involved in the splitting is lower than the gain obtained from less prediction error due to more accurate motion estimation. Not only is it optimal in the sense of bit saving, but it also eliminates the need for setting a threshold.

The number of bits needed for encoding motion information can be estimated in a straightforward manner. As far as the prediction error is concerned, the bits required can be represented by a total entropy of the prediction error, which can be estimated by using an analytical expression presented by Dufaux (1994) and Moscheni et al. (1993). Note that the coding cost for quad-tree segmentation information is negligible compared with that used for encoding prediction error and motion vectors and, hence, is omitted in determining the criterion.

FIGURE 11.18 The 20th frame of the "Flower Garden" sequence.

In addition to this entropy criterion, a more advanced procedure is adopted in the algorithm for down-projecting the motion vectors between two consecutive grids in the coarse-to-fine iterative refinement process.

Both qualitative and quantitative assessments in experiments demonstrate its good performance. It was reported that, when the PSNR is fixed, the bit rate saving for the sequence "Flower Garden" is from 10 to 20%, for "Mobile Calendar" from 6 to 12%, and for "Table Tennis" up to 8%. This can be translated into a gain in the PSNR ranging from 0.5 to 1.5 dB. Subjectively, the visual quality is improved greatly. In particular, moving edges become much sharper. Figures 11.18, 11.19, and 11.20 show a frame from "Flower Garden," "Mobile Calendar," and "Table Tennis" sequences, respectively.

11.6.3 Predictive Motion Field Segmentation

As pointed at the beginning of Section 11.5, the block-based model, which assumes constant motion within each block, leads to unreliable motion estimation and compensation. This block effect becomes more obvious and severe for motion-discontinuous areas in image frames. This is because there are two or more regions in a block in the areas, each having a different motion. Using one motion vector to represent and compensate for the whole block results in a significant prediction error increase.

Orchard (1993) proposed a predictive motion field segmentation technique to improve motion estimation and compensation along boundaries of moving objects. Significant improvement in the accuracy of the motion-compensated frame was achieved through relaxing the restrictive block-based model along moving boundaries. That is, for those blocks involving moving boundaries, the motion field assumes pixel resolution instead of block resolution.

Two key issues have to be resolved in order to realize the idea. One is the segmentation issue. It is known that the segmentation information is needed at the receiving end for motion compensation. This gives rise to a large increase in side information. To maintain almost the same amount of coding cost as the conventional block matching technique, the motion field segmentation was proposed to be conducted based on previously decoded frames. This scheme is based on the following observation: the shape of a moving object does not change from frame to frame.

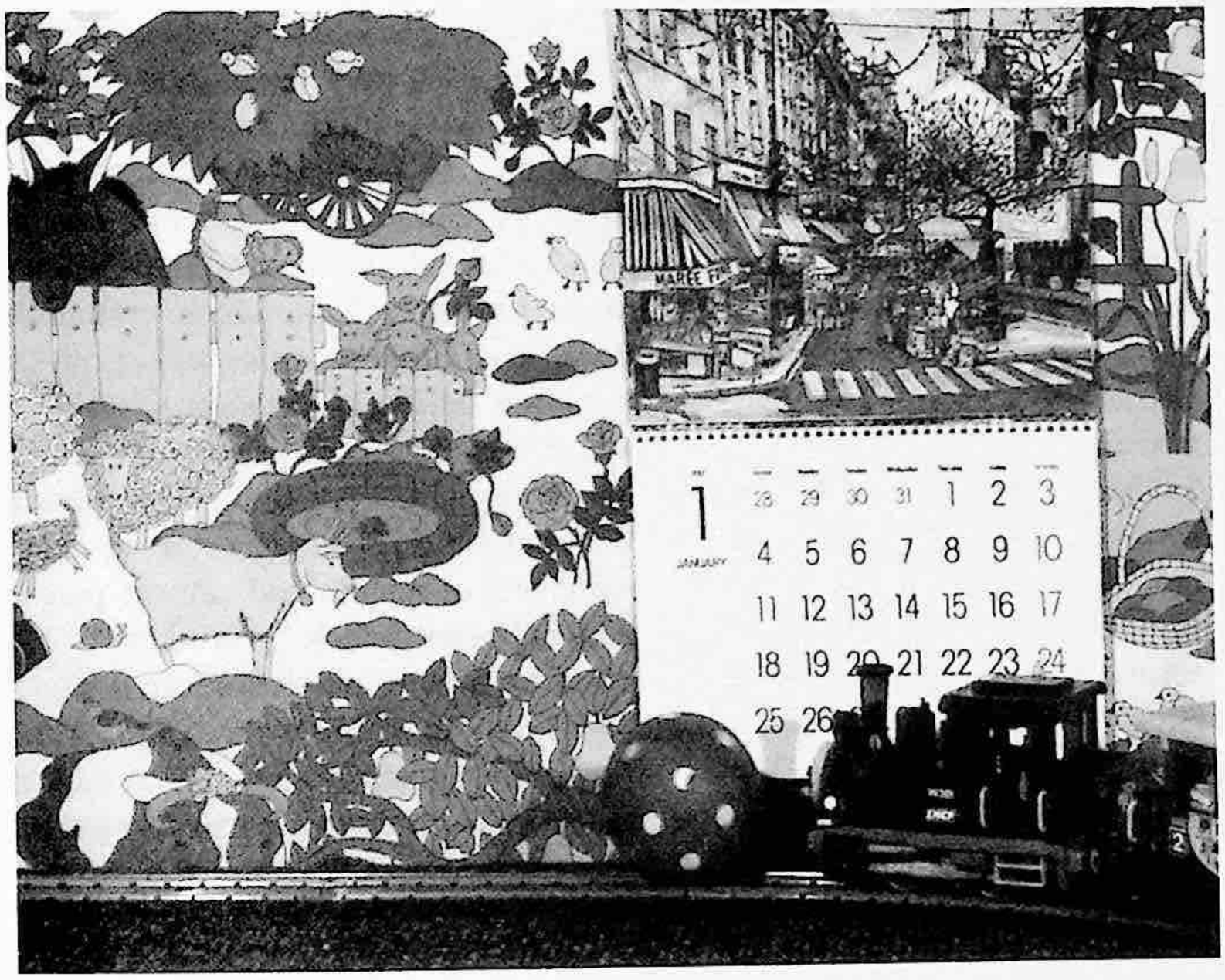


FIGURE 11.19 The 20th frame of the "Mobile and Calendar" sequence.

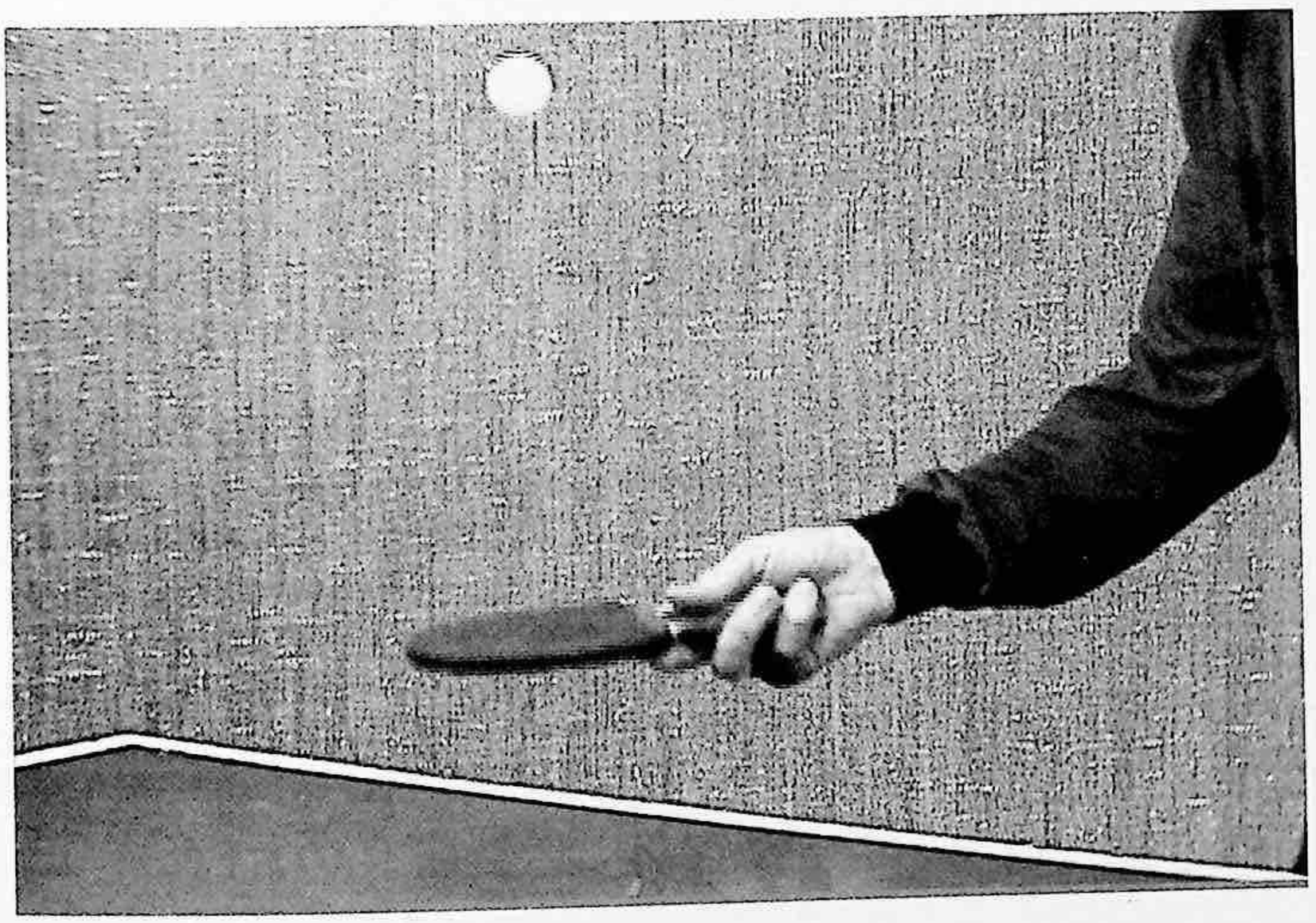


FIGURE 11.20 The 20th frame of the "Table Tennis" sequence.

This segmentation is similar to the pel recursive technique (which will be discussed in detail in the next chapter) in the sense that both techniques operate *backwards*: based on previously decoded frames. The segmentation is different from the pel recursive method in that it only uses previously decoded frames to predict the shape of discontinuity in the motion field; not the whole motion field itself. Motion vectors are still estimated using the current frame at the encoder.