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.

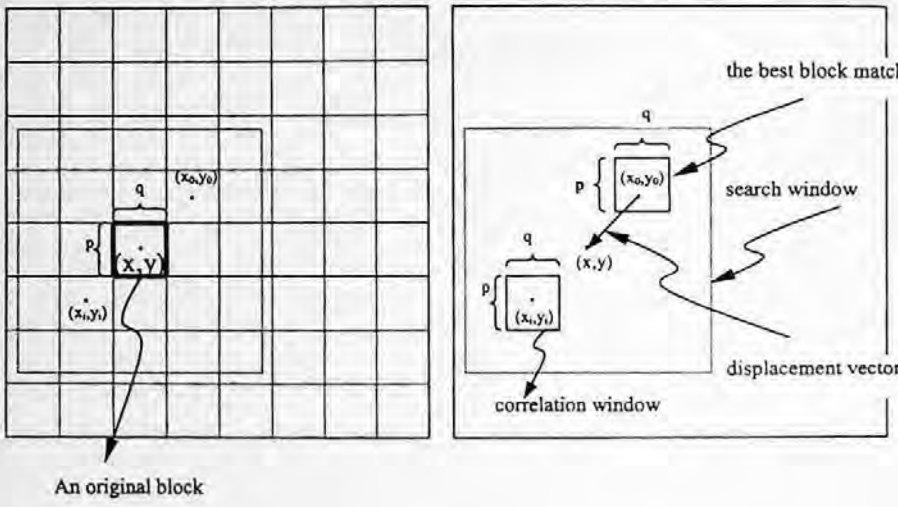
In this chapter various aspects of block matching are addressed. They include the concept and algorithm, matching criteria, searching strategies, limitations, and new improvements.

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



(a) t_n frame

(b) t_{n-1} frame

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

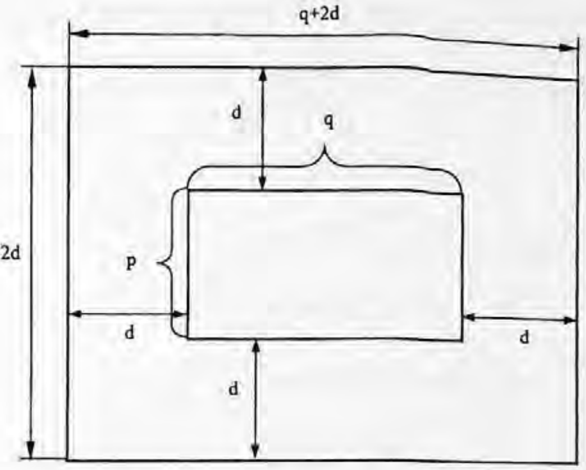


FIGURE 11.2 Search window and correlation window.

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).

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

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.

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

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.

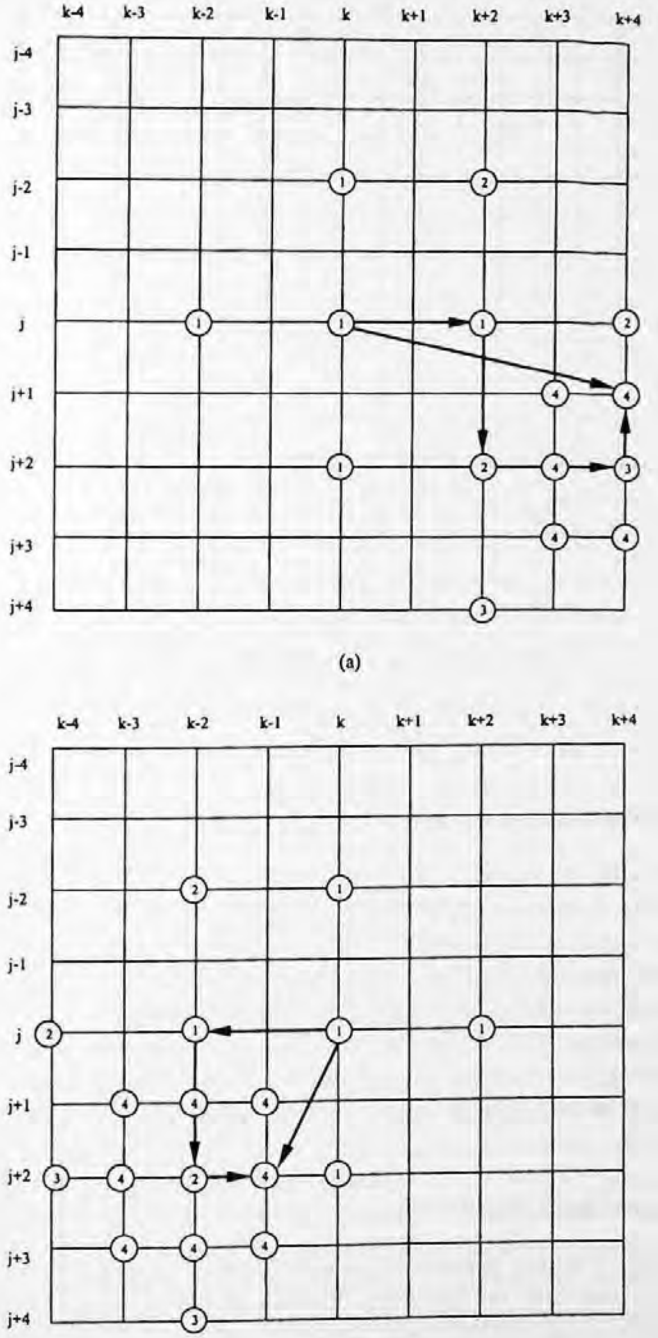
11.3.1 Full Search

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.

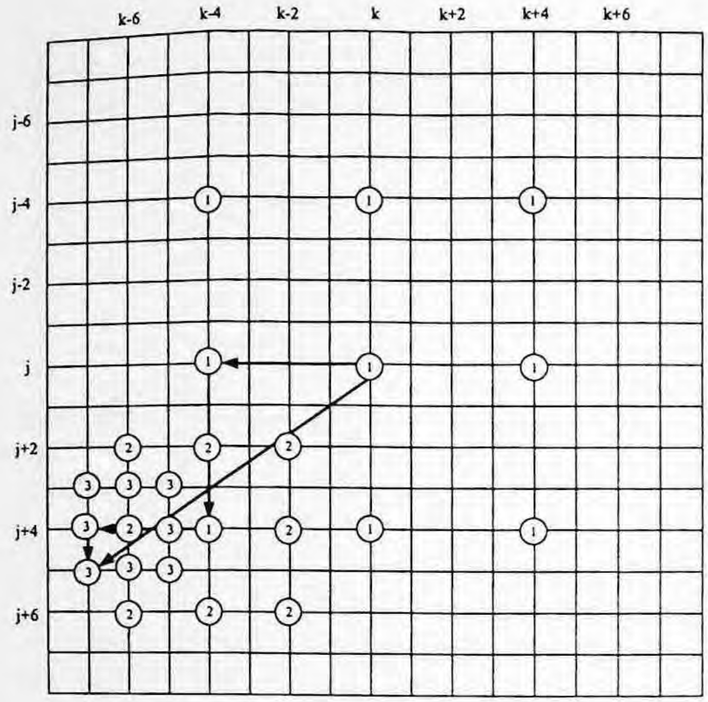


FIGURE 11.4 Three-step search procedure. Points (j+4, k-4), (j+4, k-6), and (j+5, k-7) give the minimum dissimilarity in steps 1, 2, and 3, respectively.

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),

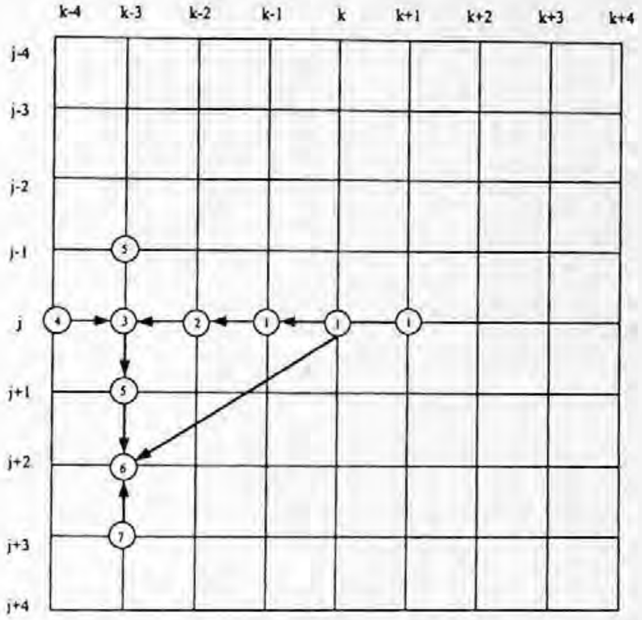


FIGURE 11.5 Conjugate direction search.

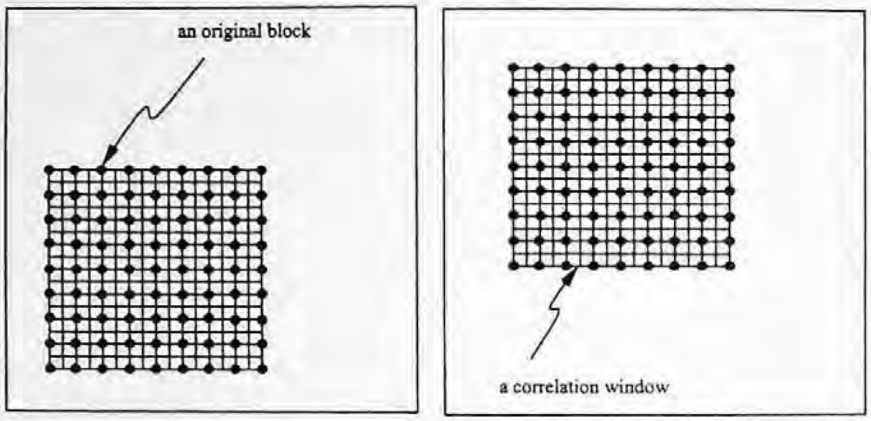
and (j, k) are examined. The procedure continues, finding point (j, k–3) as the best match along the horizontal direction since its D value is smaller than that of points (j, k–4) and (j, k–2). The procedure is then conducted along the vertical direction. In this example the best matching is finally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

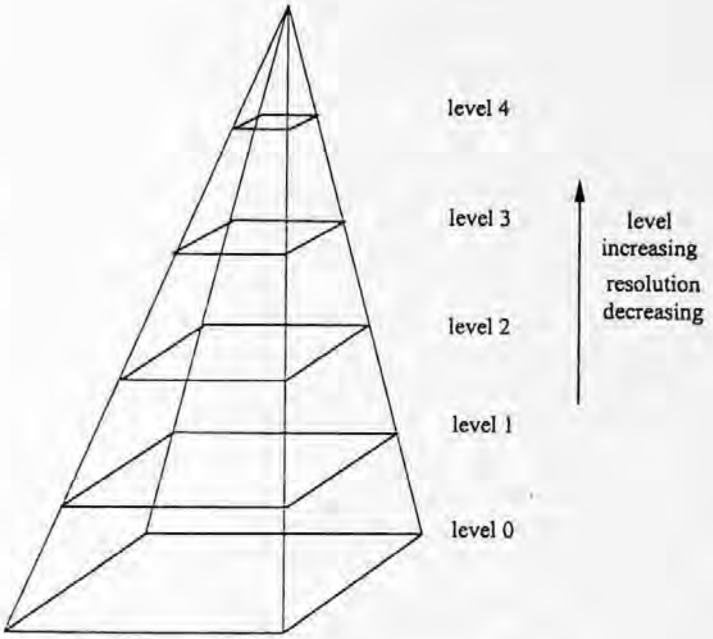


FIGURE 11.7 Gaussian pyramid structure.

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 $(\mathrm{i, j})_{\mathrm{n}}^{\mathrm{l}}$. The horizontal and vertical dimensions of a block at level l are denoted by $\mathrm{b}_{\mathrm{x}}^{\mathrm{l}}$ and $\mathrm{b}_{\mathrm{y}}^{\mathrm{l}}$, 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 $\mathrm{b}_{\mathrm{x}}^{\mathrm{l}}$, then the size of the block at level $l + 1$ becomes $2\mathrm{b}_{\mathrm{x}}^{\mathrm{l}} \times 2\mathrm{b}_{\mathrm{y}}^{\mathrm{l}}$. 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 $(\mathrm{i, j})^{\mathrm{l}}\mathrm{b}_{\mathrm{n}}^{\mathrm{l}}$ of the current frame and block $(\mathrm{i} + \mathrm{v_x}, \mathrm{j} + \mathrm{v_y})^{\mathrm{l}}\mathrm{b}_{\mathrm{n-1}}^{\mathrm{l}}$ 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).

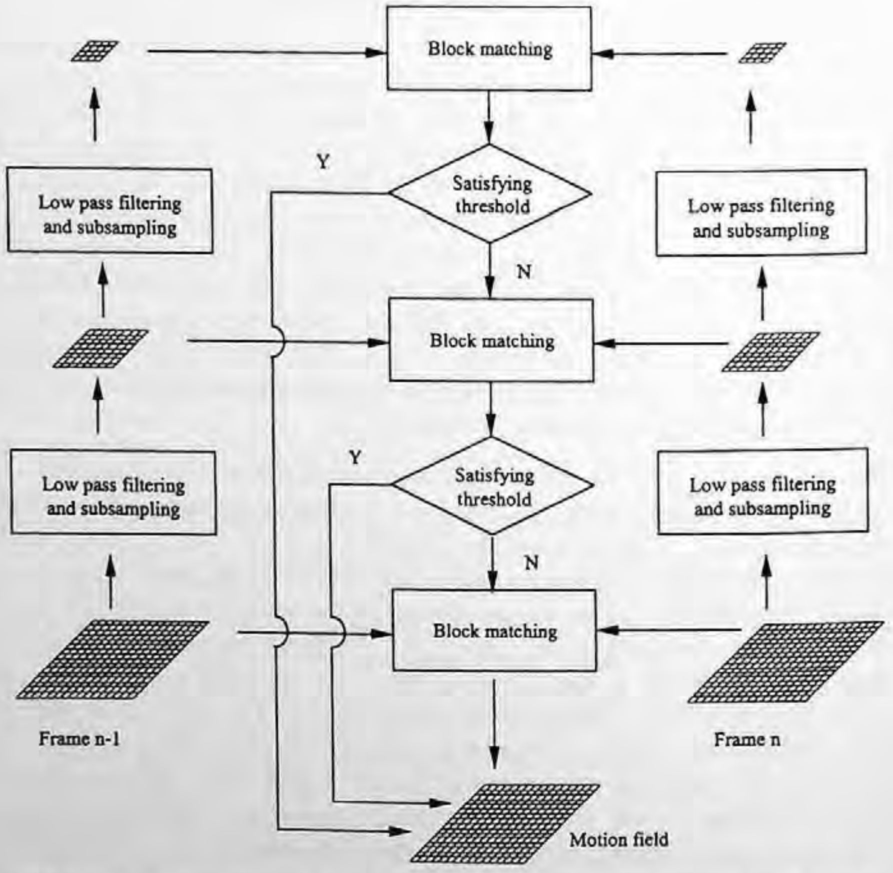

Block matching
Y
Satisfying threshold
N
Low pass filtering and subsampling
Low pass filtering and subsampling
Block matching
Y
Satisfying threshold
N
Low pass filtering and subsampling
Low pass filtering and subsampling
Block matching
Frame n-1
Frame n
Motion field

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

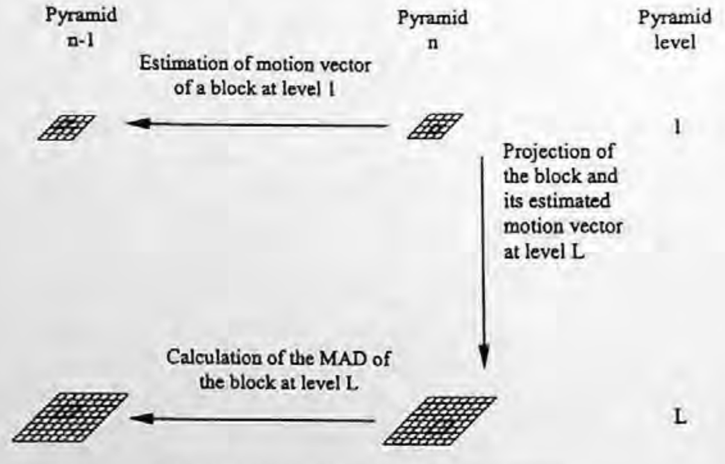


FIGURE 11.9 The thresholding process.

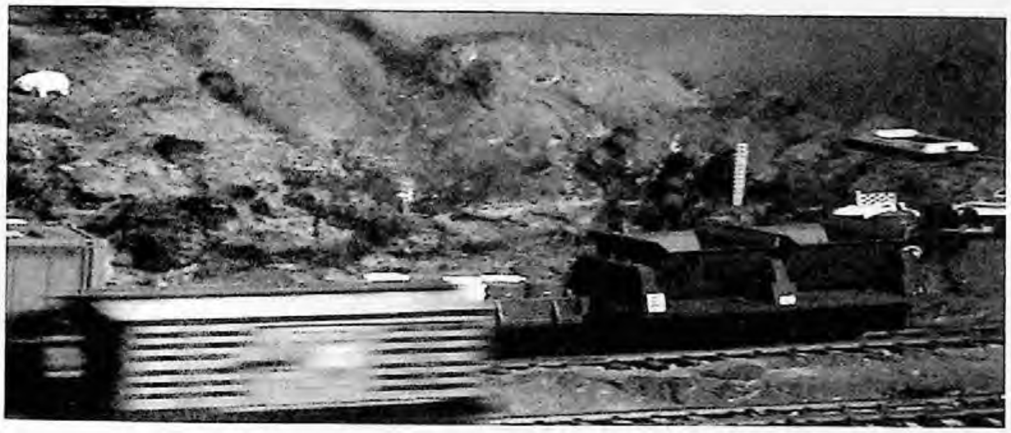


FIGURE 11.10 The 20th frame of the "Train" sequence.

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,

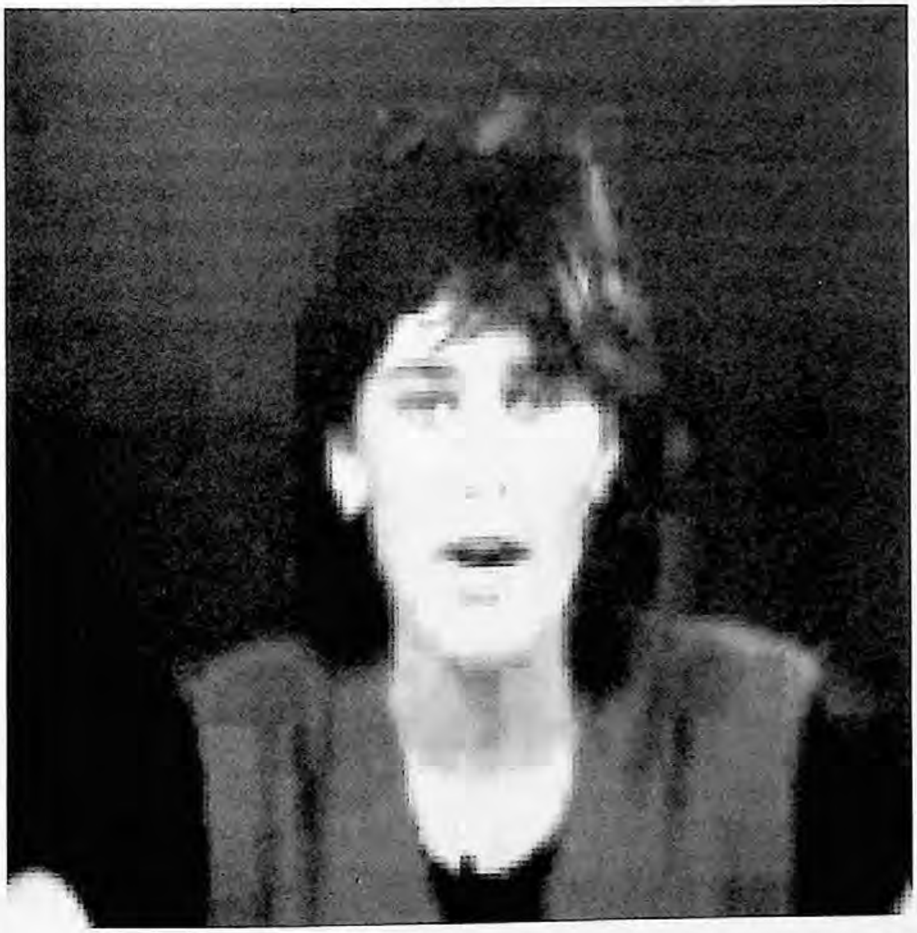


FIGURE 11.12 The 21st reconstructed frame of the "Miss America" sequence using a codec following H.263.

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).

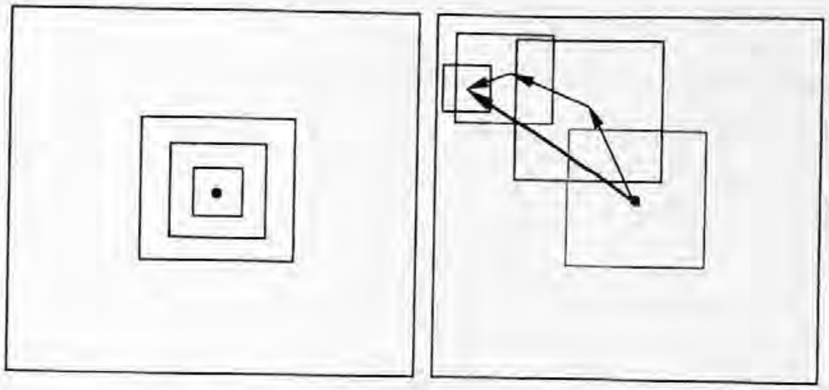
11.6 NEW IMPROVEMENTS

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).

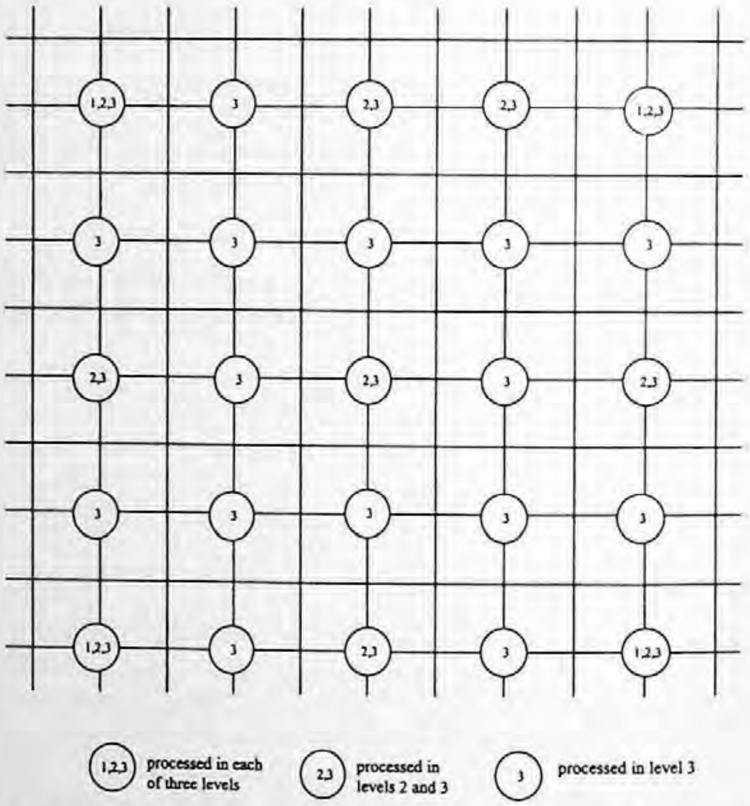


FIGURE 11.14 A portion of an image with pixels processed in all three levels.

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

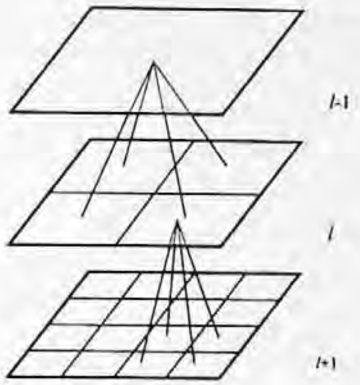


FIGURE 11.15 Illustration of a three-level hierarchical structure.

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

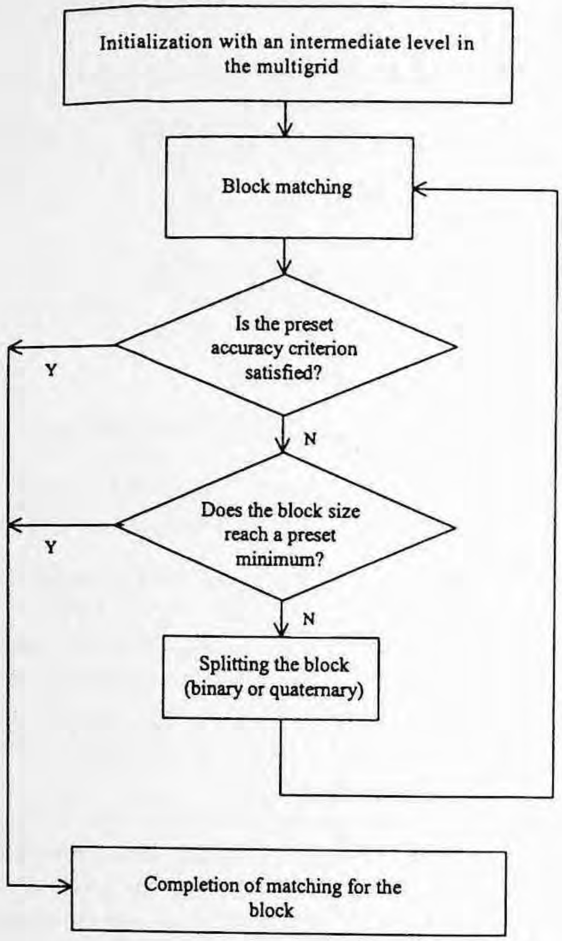

Initialization with an intermediate level in the multigrid
Block matching
Is the preset accuracy criterion satisfied?
Y
N
Does the block size reach a preset minimum?
Y
N
Splitting the block (binary or quaternary)
Completion of matching for the block

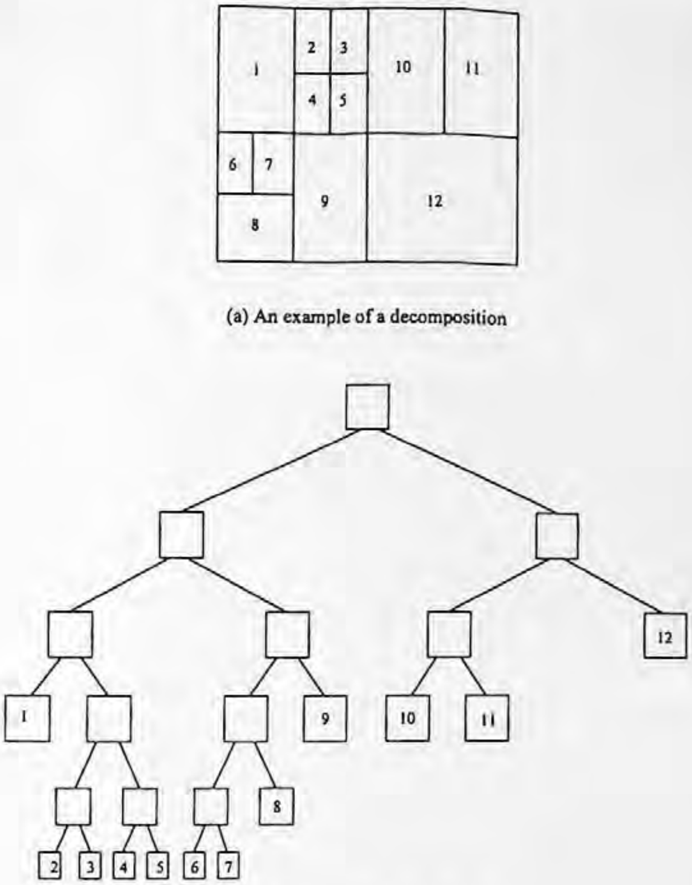
FIGURE 11.16 Flow chart of multigrid block matching.

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.

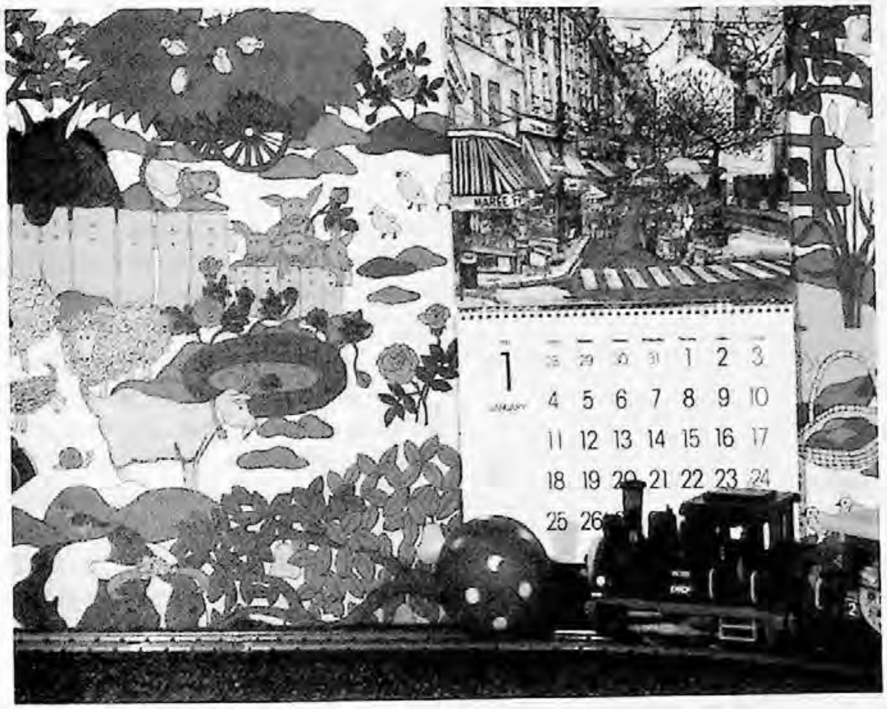


FIGURE 11.19 The 20th frame of the "Mobile and Calendar" sequence.

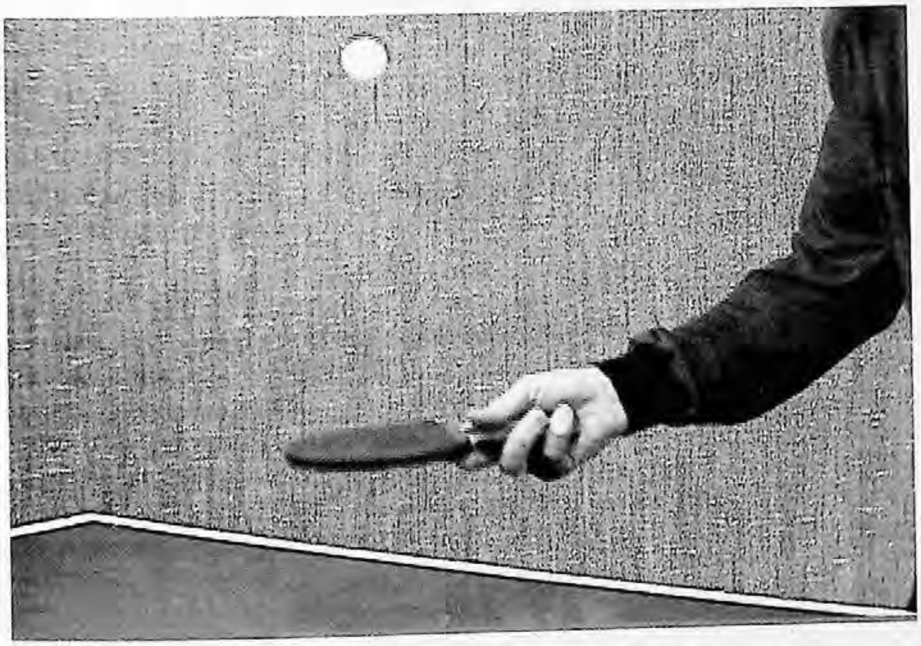


FIGURE 11.20 The 20th frame of the "Table Tennis" sequence.

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.

Consequently, this scheme is capable of achieving high accuracy in motion estimation, and at the same time it does not cause a large increase in side information due to the motion field segmentation.

Another key issue is how to achieve a reconstructed motion field with pixel resolution along moving boundaries. In order to avoid extra motion vectors that need to be encoded and transmitted, the motion vectors applied to these segmented regions in the areas of motion discontinuity are selected from a set of neighboring motion vectors. As a result, the proposed technique is capable of reconstructing discontinuities in the motion field at pixel resolution while maintaining the same amount of motion vectors as the conventional block matching technique.

A number of algorithms using this type of motion field segmentation technique have been developed and their performance has been tested and evaluated on some real video sequences (Orchard, 1993). Two of the 40-frame test sequences used were the "Table Tennis" and the "Football" sequences. The former contains fast ball motion and camera zooming, while the latter contains small objects with relatively moderate amounts of motion and camera panning. Several proposed algorithms were compared with conventional block matching in terms of average pixel prediction error energy and bits per frame required for coding prediction error. For the average pixel prediction error energy, the proposed algorithms achieve a significant reduction, ranging from –0.7 to –2.8 dB with respect to the "Table Tennis" sequence, and from –1.3 to –4.8 dB with the "Football" sequence. For bits per frame required for coding prediction error, a reduction of 20 to 30% was reported.

11.6.4 Overlapped Block Matching

All the techniques discussed so far in this section aim at more reliable motion estimation. As a result, they also alleviate annoying block artifacts to a certain extent. In this subsection we discuss a group of techniques, termed overlapped block matching, developed to alleviate or eliminate block artifacts (Watanabe, 1991; Nogaki and Ohta, 1992; Auyeung et al., 1992).

The idea is to relax the restriction of a nonoverlapped block partition imposed in the block-based model in block matching. After the nonoverlapped, fixed size, small rectangular block partition has been made, each block is enlarged along all four directions from the center of the block. Refer to Figure 11.21. Both motion estimation (block matching) and motion-compensated prediction are conducted in the same manner as that in block matching except for the inclusion of

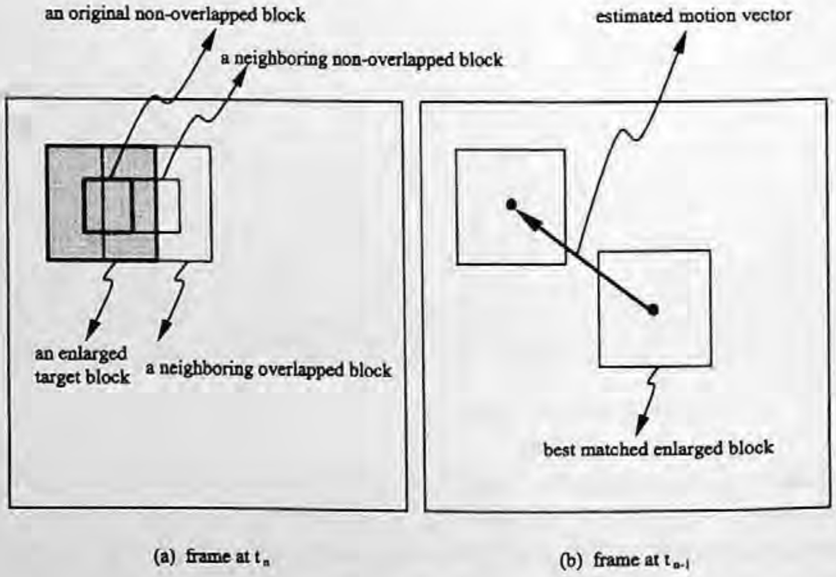

an original non-overlapped block
a neighboring non-overlapped block
estimated motion vector
an enlarged target block
a neighboring overlapped block
best matched enlarged block
(a) frame at t_n
(b) frame at t_{n-1}

FIGURE 11.21 Overlapped block matching.

a window function. That is, a 2-D window function is utilized in order to maintain an appropriate quantitative level along the overlapped portion. The window function decays towards the boundaries. In (Nogaki and Ohta, 1992) a sine-shaped window function was used.

Next, we use the algorithm proposed by Nogaki and Ohta as an example to specifically illustrate this type of technique. Consider one of the enlarged, overlapped original (also known as target) blocks, T(x,y), with a dimension of $l \times l$. Assume that a vector v_i is one of the candidate displacement vectors under consideration. The predicted version of the target block with v_i is denoted by v_i, P_{v_i} (x, y). Thus, the prediction error with v_i, E_{v_i} (x, y) can be calculated according to the following equation

$$E_{v_i}(x,y) = P_{v_i}(x,y) - T(x,y) \tag{11.8}$$

The window function W(x, y) is applied at this stage as follows, resulting in a window-operated prediction error with v_i, WE_{v_i}.

$$WE_{v_i}(x,y) = E_{v_i}(x,y) \times W(x,y) \tag{11.9}$$

Assume that the MAD is used as the matching criterion. It can then be determined as usual by using the window-operated prediction error WE_{v_i} (x, y). That is,

$$MAD = \frac{1}{l^2} \sum_{x=1}^{l} \sum_{y=1}^{l} \left| WE_{v_i}(x,y) \right|. \tag{11.10}$$

The best match, which corresponds to the minimum MAD, produces the displacement vector v.

In motion-compensated prediction, the predicted version of the enlarged target block, P_v (x, y) is derived from the frame at t_{i-1} by using estimated vector v. The same window function W (x, y) is used to generate the final window-operated predicted version of the target block. That is,

$$WP_v(x,y) = P_v(x,y) \times W(x,y) \tag{11.11}$$

It was reported by Nogaki (1992) that the luminance signal of an HDTV sequence was used in computer simulation. A block size of 16 × 16 was used for conventional block matching, while a block size of 32 × 32 was employed for the proposed overlapped block matching. The maximum displacement range d was taken as d = 15, i.e., from −15 to +15 in both the horizontal and vertical directions. The simulation indicated a reduction in the power of the prediction error by about 19%. Subjectively, it was observed that the blocking edges originally existing in the prediction error signal with conventional block matching was largely removed with the proposed overlapped block matching technique.

11.7 SUMMARY

By far, block matching is used more frequently than any other motion estimation technique in motion-compensated coding. By partitioning a frame into nonoverlapped, equally spaced, fixed size, small rectangular blocks and assuming that all the pixels in a block experience the same translational motion, block matching avoids the difficulty encountered in motion estimation of arbitrarily shaped blocks. Consequently, block matching is much simpler and involves less side information compared with motion estimation with arbitrarily shaped blocks.

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 important observation, originally made by Jain and Jain, has been confirmed again and again since then.

Various issues related to block matching such as selection of block sizes, matching criteria, search strategies, matching accuracy, and its limitations and improvements are discussed in this chapter. Specifically, a block size of 16 × 16 is used most often. For more accurate motion estimation, the size of 8 × 8 is used sometimes. In the latter case, more accurate motion estimation is obtained at the cost of more side information and higher computational complexity.

There are several different types of matching criteria that can be used in block matching. Since it was shown that the different criteria do not cause significant differences in block matching, the mean absolute difference is hence preferred due to its simplicity in implementation.

On the one hand, a full-search procedure delivers good accuracy in searching for the best match. On the other hand, it requires a large amount of computation. In order to lower computational complexity, several fast searching procedures were developed: 2-D logarithmic search, coarse-fine three-step search, and conjugate direction search, to name a few.

Besides these suboptimum search procedures, there are some other measures developed to lower computation. One of them is subsampling in the original blocks and the correlation windows. By subsampling, the computational burden in block matching can be reduced drastically, while the accuracy of the estimated motion vectors may be affected. Therefore, the subsampling procedure is only recommended for the case with a large block size.

Naturally, the multiresolution structure, a powerful computational configuration in image processing, lends itself well to a fast search in block matching. It significantly reduces the computations involved. Thresholding multiresolution block matching further saves computation.

In terms of matching accuracy, several common choices are one-pixel, half-pixel, and quarter-pixel accuracies. Spatial interpolation is usually required for half-pixel and quarter-pixel accuracies. That is, a higher accuracy is achieved with more computation.

Limitations with block matching techniques are mainly an unreliable motion vector field and block artifacts. Both are caused by the simple model: each block is assumed to experience a uniform translation. Much efforts have been made to improve these drawbacks. Several techniques that are an improvement over the conventional block matching technique are discussed in this chapter.

In the hierarchical block matching technique, a set of different sizes for both the original block and the correlation window are used. The first level in the hierarchy with a large window size and a large displacement range determines a major portion of the displacement vector reliability. The successive levels with smaller window sizes and smaller displacement ranges are capable of adaptively estimating motion vectors more locally.

The multigrid block matching technique uses multigrid structure, another powerful computational structure in image processing, to provide a variable size block matching. With a split-and-merge strategy, the thresholding multigrid block matching technique segments an image into a set of variable size blocks, each of which experiences an approximately uniform motion. A tree structure (bin-tree or quad-tree) is used to record the relationship between these variable size blocks. With the flexibility provided through the variable-size methodology, the thresholding block matching technique is capable of making the motion model of the uniform motion within each block more accurate than fixed-size block matching can do.

As pointed out in Chapter 10, the ultimate goal of motion compensation in video coding is to achieve a high coding efficiency. In other words, accurate true motion estimation is not the final goal. From this point of view, in the above-mentioned multigrid block matching, the decision of splitting a block is made only when the bits used to encode extra motion vectors involved in the splitting are less than the bits saved from encoding reduced prediction error due to more accurate estimation. To this end, an adaptive entropy criterion is proposed and used in the optimal multigrid block matching technique. Not only is it optimal in the sense of bit saving, but it also eliminates the need for setting a threshold.

Apparently the block-based model encounters a more severe problem along moving boundaries. To solve the problem, the predictive motion field segmentation technique make the blocks involving moving boundaries have the motion field measured with pixel resolution instead of block resolution. In order to save shape overhead, segmentation is carried out backwards, i.e., based on previously decoded frames. In order to avoid a large increase of side information associated with extra motion vectors, the motion vectors applied to these segmented regions along moving boundaries are selected from a set of neighboring motion vectors. As a result, the technique is capable of reconstructing discontinuities in the motion field at pixel resolution while maintaining the same amount of motion vectors as the conventional block matching technique.

The last improvement over conventional block matching discussed in this chapter is overlapped block matching. In contrast to dealing with blocks independently of each other, the overlapped block matching technique enlarges blocks so as to make them overlap. A window function is then constructed and used in both motion estimation and motion compensation. Because it relaxes the restriction of a nonoverlapped block partition imposed by conventional block matching, it achieves better performance than the conventional block matching.

11.8 EXERCISES

11-1. Refer to Figure 11.2. It is said that there are a total of $(2d + 1) \times (2d + 1)$ positions that need to be examined in block matching with full search if one-pixel accuracy is required. How many positions are there that need to be exmined in block matching with full search if half-pixel and quarter-pixel accuracies are required?

11-2. What are the two effects that subsampling in the original block and the correlation block may bring out?

11-3. Read Burt and Adelson (1983) or Burt (1984), and explain why the pyramid is named after Gauss.

11-4. Read Burt and Adelson (1983) or Burt (1984), and explain why a pyramid structure is considered as a powerful computational configuration. Specifically, in multiresolutional block matching, how and to what extent does it save computation dramatically, compared with the conventional block matching technique? You may want to refer to Section 11.3.7.

11-5. How is the threshold determined in the thresholding multidimensional block matching technique (refer to Section 11.3.7). It is said that the square root of the MSE value, derived from the given PSNR according to Equation 11.6, is used as an initial threshold value. Justify the necessity of the square root operation.

11-6. Refer to Section 11.6.1 or the paper by Bierling (1988). State the different requirements in the applications of motion-compensated interpolation and motion-compensated coding. Discuss where a full resolution of the translational motion vector field may be used?

11-7. Read the paper Dufaux and Moscheni (1995), and explain the main feature of optimal multigrid block matching. State how the adaptive entropy criterion is established. Implement the algorithm and compare its performance with that presented by Chan et al. (1990).

11-8. Learn the predictive motion field segmentation technique (Orchard, 1993). Explain how the algorithms avoid a large increase in overhead due to motion field segmentation.

11-9. Implement the overlapped block matching algorithm introduced by Nogaki (1992). Compare its performance with that of the conventional block matching technique.

REFERENCES

Anandan, P. Measurement Visual Motion From Image Sequences, Ph.D. thesis, COINS Department, University of Massachusetts, Amherst, 1987.

Anuta, P. F. Digital registration of multispectral video imagery, *Soc. Photo-Opt. Instrum. Eng. J.*, 7, 168-175, 1969.

Auyeung, C., J. Kosmach, M. Orchard, and T. Kalafatis, Overlapped block motion compensation, *SPIE Proc. Visual Commun. Image Process. '92*, Boston, MA, Nov. 1992, vol. 1818, 561-571.

Bierling, M. Displacement estimation by hierarchical blockmatching, *SPIE Proc. Visual Commun. Image Process.*, 1001, 942-951, 1988.

Brailean, J. Universal Accessibility and Object-Based Functionality, *ISCAS Tutorial on MPEG 4*, June 1997, Chap. 3.3.

Brofferio, S. and F. Rocca, Interframe redundancy reduction of video signals generated by translating objects, *IEEE Trans. Commun.*, COM-25, 448-455, 1977.

Burt, P. J. and E. H. Adelson, The Laplacian pyramid as a compact image code, *IEEE Trans. Commun.*, COM-31(4), 532-540, 1983.

Burt, P. J. The pyramid as a structure for efficient computation, in *Multiresolution Image Processing and Analysis*, A. Rosenfeld, Ed., Springer-Verlag, New York, 1984, 6.

Cafforio, C. and F. Rocca, Method for measuring small displacement of television images, *IEEE Trans. Inf. Theory*, IT-22, 573-579, 1976.

Chan, M. H., Y. B. Yu, and A. G. Constantinides, Variable size block matching motion compensation with applications to video coding, *IEEE Proc.*, 137(4), 205-212, 1990.

Dufaux, F. and M. Kunt, Multigrid block matching motion estimation with an adaptive local mesh refinement, *SPIE Proc. Visual Commun. Image Process. '92*, 1818, 97-109, 1992.

Dufaux, F. Multigrid Block Matching Motion Estimation for Generic Video Coding, Ph.D. dissertation, Swiss Federal Institute of Technology, Lausanne, Switzerland, 1994.

Dufaux, F. and F. Moscheni, Motion estimation techniques for digital TV: A review and a new contribution, *Proc. IEEE*, 83(6), 858-876, 1995.

Hackbusch, W. and U. Trottenberg, Eds., *Multigrid Methods*, Springer-Verlag, New York, 1982.

Haskell, B. G. and J. O. Limb, Predictive video encoding using measured subject velocity, U.S. Patent 3,632,865, January 1972.

Jain, J. R. and A. K. Jain, Displacement measurement and its application in interframe image coding, *IEEE Trans. Commun.*, COM-29(12), 1799-1808, 1981.

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

Koga, T., K. Linuma, A. Hirano, Y. Iijima, and T. Ishiguro, Motion-compensated interframe coding for video conferencing, *Proc. NTC'81*, G5.3.1-G5.3.5, New Orleans, LA, Dec. 1981.

Knuth, D. E. Searching and Sorting, *The Art of Computer Programming*, Vol. 3, Addison-Wesley, Reading, MA, 1973.

Limb, J. O. and J. A. Murphy, Measuring the speed of moving objects from television signals, *IEEE Trans. Commun.*, COM-23, 474-478, 1975.

Lin, S., Y. Q. Shi, and Y.-Q. Zhang, An optical flow-based motion compensation algorithm for very low bit-rate video coding, *Proc. 1997 IEEE Int. Conf. Acoustics, Speech Signal Process.*, 2869-2872, Munich, Germany, April 1997; *Int. J. Imaging Syst. Technol.*, 9(4), 230-237, 1998.

Moscheni, F., F. Dufaux, and H. Nicolas, Entropy criterion for optimal bit allocation between motion and prediction error information, in *SPIE 1993 Proc. Visual Commun. Image Process.*, 235-242, Cambridge, MA, Nov. 1993.

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

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

Nogaki, S. and Ohta, M., An overlapped block motion compensation for high-quality motion picture coding, *Proc. IEEE Int. Symp. Circuits and Systems*, vol. 1, 184-187, San Diego, 1992.

Orchard, M. T. Predictive motion-field segmentation for image sequence coding, *IEEE Trans. Circuits and Syst. Video Technol.*, 3(1), 54-69, 1993.

Pratt, W. K. Correlation techniques of image registration, *IEEE Trans. Aerosp. Electron. Syst.*, AES-10(3), 353-358, 1974.

Rocca, F. and Zanoletti, S., Bandwidth reduction via movement compensation on a model of the random video process, *IEEE Trans. Comm.*, vol. COM-20, 960-965, Oct. 1972.

Shi, Y. Q. and X. Xia, A thresholding multidimensional block matching algorithm, *IEEE Trans. Circuits and Syst. Video Technol.*, 7(2), 437-440, April 1997.

Srinivasan, R. and K. R. Rao, Predictive coding based on efficient motion estimation, *Proc. of ICC*, 521-526, May 1984.

Tzovaras, D., M. G. Strintzis, and H. Sahinolou, Evaluation of multiresolution block matching techniques for motion and disparity estimation, *Signal Process. Image Commun.*, 6, 56-67, 1994.

Watanabe, H. and Singhal, S., Windowed motion compensation, SPIE, vol. 1605, in *Visual Communications and Image Processing, 1991: Visual Communication,* 582-589, November 1991.

Xia, X. and Y. Q. Shi, A thresholding hierarchical block matching algorithm, *Proc. IEEE 1996 Int. Symp. Circuits Syst.*, II, pp. 624-627, Atlanta, GA, May 1996; *J. Comput. Sci. Inf. Manage.*, 1(2), 83-90, 1998.

12 Pel Recursive Technique

As discussed in Chapter 10, the pel recursive technique is one of the three major approaches to two-dimensional displacement estimation in image planes for the signal processing community. Conceptually speaking, it is one type of region-matching technique. In contrast to block matching (which was discussed in the previous chapter), it *recursively* estimates displacement vectors for *each pixel* in an image frame. The displacement vector of a pixel is estimated by recursively minimizing a nonlinear function of the dissimilarity between two certain regions located in two consecutive frames. Note that *region* means a group of pixels, but it could be as small as a single pixel. Also note that the terms *pel* and *pixel* have the same meaning. Both terms are used frequently in the field of signal and image processing.

This chapter is organized as follows. A general description of the recursive technique is provided in Section 12.1. Some fundamental techniques in optimization are covered in Section 12.2. Section 12.3 describes the Netravali and Robbins algorithm, the pioneering work in this category. Several other typical pel recursive algorithms are introduced in Section 12.4. In Section 12.5, a performance comparison between these algorithms is made.

12.1 PROBLEM FORMULATION

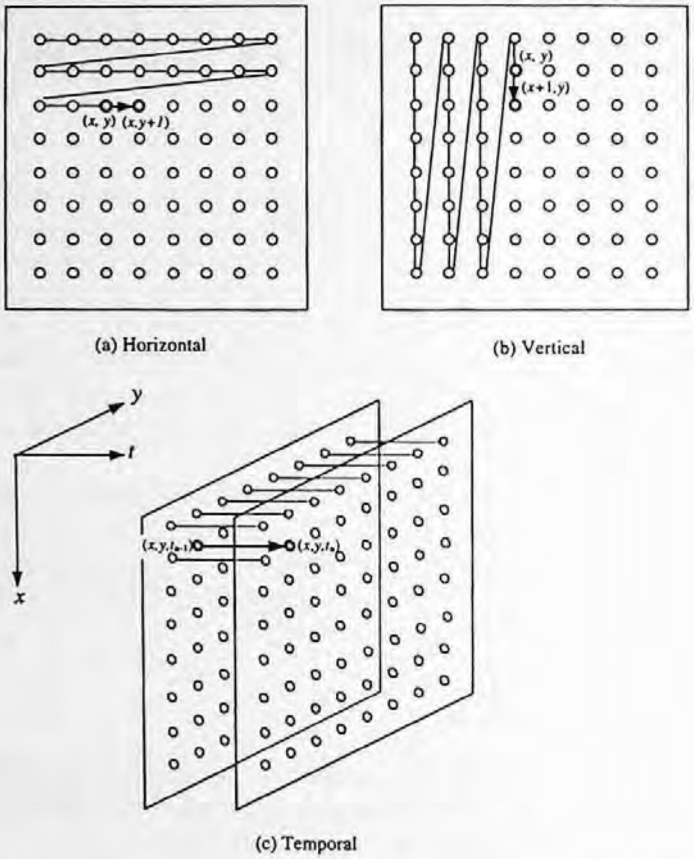
In 1979 Netravali and Robbins published the first pel recursive algorithm to estimate displacement vectors for motion-compensated interframe image coding. Netravali and Robbins (1979) defined a quantity, called the displaced frame difference (DFD), as follows.

$$DFD\left(x,y;d_x,d_y\right)=f_n(x,y)-f_{n-1}\left(x-d_x,y-d_y\right), \tag{12.1}$$

where the subscript n and $n-1$ indicate two moments associated with two successive frames based on which motion vectors are to be estimated; x, y are coordinates in image planes, d_x, d_y are the two components of the displacement vector, $\vec{d}$, along the horizontal and vertical directions in the image planes, respectively. $DFD(x, y; d_x, d_y)$ can also be expressed as $DFD(x, y; \vec{d}$. Whenever it does not cause confusion, it can be written as DFD for the sake of brevity. Obviously, if there is no error in the estimation, i.e., the estimated displacement vector is exactly equal to the true motion vector, then DFD will be zero.

A nonlinear function of the DFD was then proposed as a dissimilarity measure by Netravali and Robbins (1979), which is a square function of DFD, i.e., DFD^2.

Netravali and Robbins thus converted displacement estimation into a minimization problem. That is, each pixel corresponds to a pair of integers (x, y), denoting its spatial position in the image plane. Therefore, the DFD is a function of $\vec{d}$. The estimated displacement vector $\vec{d} = (d_x, d_y)^T$, where $(\ )^T$ denotes the transposition of the argument vector or matrix, can be determined by minimizing the DFD^2. This is a typical nonlinear programming problem, on which a large body of research has been reported in the literature. In the next section, several techniques that rely on a method, called descent method, in optimization are introduced. The Netravali and Robbins algorithm can be applied to a pixel once or iteratively applied several times for displacement estimation. Then the algorithm moves to the next pixel. The estimated displacement vector of a pixel can be used as an initial estimate for the next pixel. This recursion can be carried out



(a) Horizontal

(b) Vertical

(c) Temporal

FIGURE 12.1 Three types of recursions: (a) horizontal; (b) vertical; (c) temporal.

horizontally, vertically, or temporally. By *temporally*, we mean that the estimated displacement vector can be passed to the pixel of the same spatial position within image planes in a temporally neighboring frame. Figure 12.1 illustrates these three different types of recursion.

12.2 DESCENT METHODS

Consider a nonlinear real-valued function z of a vector variable $\bar{x}$,

$$z = f(\bar{x}), \tag{12.2}$$

with $\bar{x} \in R^n$, where R^n represents the set of all n-tuples of real numbers. The question we face now is how to find such a vector denoted by $\bar{x}^*$ that the function z is minimized. This is classified as an unconstrained nonlinear programming problem.

12.2.1 First-Order Necessary Conditions

According to the optimization theory, if $f(\bar{x})$ has continuous first-order partial derivatives, then the first-order necessary conditions that $\bar{x}^*$ has to satisfy are

$$\nabla f(\bar{x}^*) = 0, \tag{12.3}$$

where ∇ denotes the gradient operation with respect to $\bar{x}$ evaluated at $\bar{x}^*$. Note that whenever there is only one vector variable in the function z to which the gradient operator is applied, the sign ∇ would remain without a subscript, as in Equation 12.3. Otherwise, i.e., if there is more than one vector variable in the function, we will explicitly write out the variable, to which the gradient operator is applied, as a subscript of the sign ∇. In the component form, Equation 12.3 can be expressed as

$$\begin{cases} \dfrac{\partial f(\bar{x})}{\partial x_1} = 0 \\ \dfrac{\partial f(\bar{x})}{\partial x_2} = 0 \\ \vdots \\ \dfrac{\partial f(\bar{x})}{\partial x_n} = 0. \end{cases} \tag{12.4}$$

12.2.2 Second-Order Sufficient Conditions

If $F(\bar{x})$ has second-order continuous derivatives, then the second-order sufficient conditions for $F(\bar{x}^*)$ to reach the minimum are known as

$$\nabla f(\bar{x}^*) = 0 \tag{12.5}$$

and

$$\mathbf{H}(\bar{x}^*) > 0, \tag{12.6}$$

where **H** denotes the Hessian matrix and is defined as follows.

$$\mathbf{H}(\bar{x}) = \begin{bmatrix} \dfrac{\partial^2 f(\bar{x})}{\partial^2 x_1} & \dfrac{\partial^2 f(\bar{x})}{\partial x_1 \partial x_2} & \cdots & \dfrac{\partial^2 f(\bar{x})}{\partial x_1 \partial x_n} \\ \dfrac{\partial^2 f(\bar{x})}{\partial x_2 \partial x_1} & \dfrac{\partial^2 f(\bar{x})}{\partial^2 x_2} & \cdots & \dfrac{\partial^2 f(\bar{x})}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \vdots & \vdots \\ \dfrac{\partial^2 f(\bar{x})}{\partial x_n \partial x_1} & \dfrac{\partial^2 f(\bar{x})}{\partial x_n \partial x_2} & \cdots & \dfrac{\partial^2 f(\bar{x})}{\partial^2 x_n} \end{bmatrix}. \tag{12.7}$$

We can thus see that the Hessian matrix consists of all the second-order partial derivatives of f with respect to the components of $\bar{x}$. Equation 12.6 means that the Hessian matrix **H** is positive definite.

12.2.3 Underlying Strategy

Our aim is to derive an iterative procedure for the minimization. That is, we want to find a sequence

$$\bar{x}_0, \bar{x}_1, \bar{x}_2, \cdots, \bar{x}_n, \cdots, \tag{12.8}$$

such that

$$f(\bar{x}_0) > f(\bar{x}_1) > f(\bar{x}_2) > \cdots > f(\bar{x}_n) > \cdots \tag{12.9}$$

and the sequence converges to the minimum of $f(\bar{x})$, $f(\bar{x}^*)$.

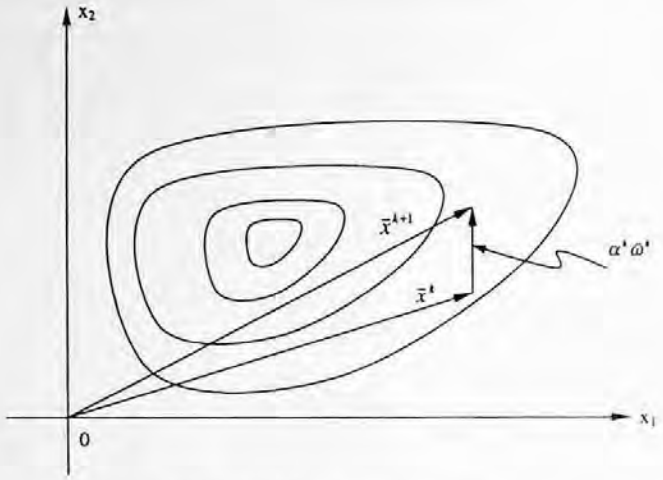


FIGURE 12.2 Descent method.

A fundamental underlying strategy for almost all the descent algorithms (Luenberger, 1984) is described next. We start with an initial point in the space; we determine a direction to move according to a certain rule; then we move along the direction to a relative minimum of the function z. This minimum point becomes the initial point for the next iteration.

This strategy can be better visualized using a 2-D example, shown in Figure 12.2. There, $\bar{x} = (x_1, x_2)^t$. Several closed curves are referred to as *contour curves* or *level curves*. That is, each of the curves represents

$$f(x_1, x_2) = c, \tag{12.10}$$

with c being a constant.

Assume that at the kth iteration, we have a guess: $\bar{x}^k$. For the $(k + 1)$th iteration, we need to

- Find a search direction, pointed by a vector $\bar{\omega}^k$;
- Determine an optimal step size α^k with $\alpha^k > 0$,

such that the next guess $\bar{x}^{k+1}$ is

$$\bar{x}^{k+1} = \bar{x}^k + \alpha^k \bar{\omega}^k \tag{12.11}$$

and $\bar{x}^{k+1}$ satisfies $f(\bar{x}^k) > f(\bar{x}^{k+1})$.

In Equation 12.11, $\bar{x}^k$ can be viewed as a prediction vector for $\bar{x}^{k+1}$, while $\alpha^k\bar{\omega}^k$ an update vector, $\bar{v}^k$. Hence, using the Taylor series expansion, we can have

$$f(\bar{x}^{k+1}) = f(\bar{x}^k) + \left\langle \nabla f(\bar{x}^k), \alpha^k\bar{\omega}^k \right\rangle + \varepsilon, \tag{12.12}$$

where $\langle s, t \rangle$ denotes the inner product between vectors $\bar{s}$ and $\bar{t}$; and ε represents the higher-order terms in the expansion. Consider that the increment of $\alpha^k\bar{\omega}^k$ is small enough and, thus, ε can be ignored. From Equation 12.10, it is obvious that in order to have $f(\bar{x}^{k+1}) < F(\bar{x}^k)$ we must have $\langle \nabla f(\bar{x}^k), \alpha^k\bar{\omega}^k \rangle < 0$. That is,

$$f(\bar{x}^{k+1}) < f(\bar{x}^k) \Rightarrow \left\langle \nabla f(\bar{x}^k), \alpha^k\bar{\omega}^k \right\rangle < 0. \tag{12.13}$$

Choosing a different update vector, i.e., the product of the $\bar{\omega}^k$ vector and the step size α^k, results in a different algorithm in implementing the descent method.

In the same category of the descent method, a variety of techniques have been developed. The reader may refer to Luenberger (1984) or the many other existing books on optimization. Two commonly used techniques of the descent method are discussed below. One is called the steepest descent method, in which the search direction represented by the $\bar{\omega}$ vector is chosen to be opposite to that of the gradient vector, and a real parameter of the step size α^k is used; the other is the Newton–Raphson method, in which the update vector in estimation, determined jointly by the search direction and the step size, is related to the Hessian matrix, defined in Equation 12.7. These two techniques are further discussed in Sections 12.2.5 and 12.2.6, respectively.

12.2.4 Convergence Speed

Speed of convergence is an important issue in discussing the descent method. It is utilized to evaluate the performance of different algorithms.

Order of Convergence — Assume a sequence of vectors $\{\bar{x}^k\}$, with $k = 0, 1, \cdots, \infty$, converges to a minimum denoted by $\bar{x}^*$. We say that the convergence is of order p if the following formula holds (Luenberger, 1984):

$$0 \le \overline{\lim}_{k\to\infty} \frac{\left|\bar{x}^{k+1} - \bar{x}^*\right|}{\left|\bar{x}^k - \bar{x}^*\right|^p} < \infty, \tag{12.14}$$

where p is positive, $\overline{\lim}$ denotes the limit superior, and $|\ |$ indicates the magnitude or norm of a vector argument. For the two latter notions, more descriptions follow.

The concept of the limit superior is based on the concept of supremum. Hence, let us first discuss the supremum. Consider a set of real numbers, denoted by Q, that is bounded above. Then there must exist a smallest real number o such that for all the real numbers in the set Q, i.e., $q \in Q$, we have $q \le o$. This real number o is referred to as the least upper bound or the supremum of the set Q, and is denoted by

$$\sup\{q : q \in Q\} \quad \text{or} \quad \sup_{q\in Q}(q). \tag{12.15}$$

Now turn to a real bounded above sequence r^k, $k = 0,1,\cdots,\infty$. If $s^k = \sup\{r^j : j \ge k\}$, then the sequence $\{s^k\}$ converges to a real number s^*. This real number s^* is referred to as the limit superior of the sequence $\{r^k\}$ and is denoted by

$$\overline{\lim}_{k\to\infty}\left(r^k\right). \tag{12.16}$$

The magnitude or norm of a vector $\bar{x}$, denoted by $|\bar{x}|$, is defined as

$$|\bar{x}| = \langle \bar{x}, \bar{x} \rangle, \tag{12.17}$$

where $\langle s, t \rangle$ is the inner product between the vector $\bar{s}$ and $\bar{t}$. Throughout this discussion, when we say *vector* we mean column vector. (Row vectors can be handled accordingly.) The inner product is therefore defined as

$$\langle \bar{s}, \bar{t} \rangle = \bar{s}, \bar{t}^{\,T}, \tag{12.18}$$

with the superscript T indicating the transposition operator.

With the definitions of the limit superior and the magnitude of a vector introduced, we are now in a position to understand easily the concept of the order of convergence defined in Equation 12.14. Since the sequences generated by the descent algorithms behave quite well in general (Luenberger, 1984), the limit superior is rarely necessary. Hence, roughly speaking, instead of the limit superior, the limit may be used in considering the speed of convergence.

Linear Convergence — Among the various orders of convergence, the order of unity is of importance, and is referred to as linear convergence. Its definition is as follows. If a sequence $\{\bar{x}^k\}$, $k = 0,1,\cdots,\infty$, converges to $\bar{x}^*$ with

$$\lim_{k\to\infty} \frac{\left|\bar{x}^{k+1} - \bar{x}^*\right|}{\left|\bar{x}^k - \bar{x}^*\right|} = \gamma < 1, \tag{12.19}$$

then we say that this sequence converges linearly with a convergence ratio γ. The linear convergence is also referred to as geometric convergence because a linear convergent sequence with convergence ratio γ converges to its limit at least as fast as the geometric sequences $c\gamma^k$, with c being a constant.

12.2.5 Steepest Descent Method

The steepest descent method, often referred to as the gradient method, is the oldest and simplest one among various techniques in the descent method. As Luenberger pointed out in his book, it remains the fundamental method in the category for the following two reasons. First, because of its simplicity, it is usually the first method attempted for solving a new problem. This observation is very true. As we shall see soon, when handling the displacement estimation as a nonlinear programming problem in the pel recursive technique, the first algorithm developed by Netravali and Robbins is essentially the steepest descent method. Second, because of the existence of a satisfactory analysis for the steepest descent method, it continues to serve as a reference for comparing and evaluating various newly developed and more advanced methods.

Formula — In the steepest descent method, $\bar{\omega}^k$ is chosen as

$$\bar{\omega} = -\nabla f\left(\bar{x}^k\right), \tag{12.20}$$

resulting in

$$f\left(\bar{x}^{k+1}\right) = f\left(\bar{x}^k\right) - \alpha^k \nabla f\left(\bar{x}^k\right), \tag{12.21}$$

where the step size α^k is a real parameter, and, with our rule mentioned before, the sign ∇ here denotes a gradient operator with respect to $\bar{x}^k$. Since the gradient vector points to the direction along which the function $f(\bar{x})$ has greatest increases, it is naturally expected that the selection of the negative direction of the gradient as the search direction will lead to the steepest descent of $f(\bar{x})$. This is where the term *steepest descent* originated.

Convergence Speed — It can be shown that if the sequence $\{\bar{x}\}$ is bounded above, then the steepest descent method will converge to the minimum. Furthermore, it can be shown that the steepest descent method is linear convergent.

Selection of Step Size — It is worth noting that the selection of the step size α^k has significant influence on the performance of the algorithm. In general, if it is small, it produces an accurate

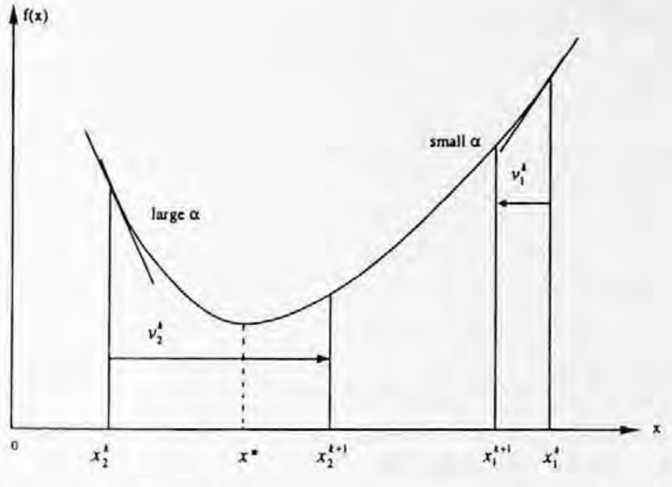


FIGURE 12.3 An illustration of effect of selection of step size on minimization performance. Too small α requires more steps to reach x^*. Too large α may cause overshooting.

estimate of $\bar{x}^*$. But a smaller step size means it will take longer for the algorithm to reach the minimum. Although a larger step size will make the algorithm converge faster, it may lead to an estimate with large error. This situation can be demonstrated in Figure 12.3. There, for the sake of an easy graphical illustration, $\bar{x}$ is assumed to be one dimensional. Two cases of too small (with subscript 1) and too large (with subscript 2) step sizes are shown for comparison.

12.2.6 Newton-Raphson's Method

The Newton–Raphson method is the next most popular method among various descent methods.

Formula — Consider $\bar{x}^k$ at the kth iteration. The k + 1th guess, $\bar{x}^{k+1}$, is the sum of $\bar{x}^k$ and $\bar{v}^k$,

$$\bar{x}^{k+1} = \bar{x}^k + \bar{v}^k, \tag{12.22}$$

where $\bar{v}^k$ is an update vector as shown in Figure 12.4. Now expand the $\bar{x}^{k+1}$ into the Taylor series explicitly containing the second-order term.

$$f(\bar{x}^{k+1}) - f(\bar{x}^k) + \langle \nabla f, \bar{v} \rangle + \frac{1}{2} \langle H(\bar{x}^k)\bar{v}, \bar{v} \rangle + \varphi, \tag{12.23}$$

where φ denotes the higher-order terms, ∇ the gradient, and **H** the Hessian matrix. If $\bar{v}$ is small enough, we can ignore the φ. According to the first-order necessary conditions for $\bar{x}^{k+1}$ to be the minimum, discussed in Section 12.2.1, we have

$$\nabla_{\bar{v}} f(\bar{x}^k + \bar{v}) = \nabla f(\bar{x}^k) + \mathbf{H}(\bar{x}^k)\bar{v} = 0, \tag{12.24}$$

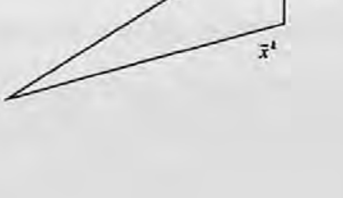


FIGURE 12.4 Derivation of the Newton–Raphson method.

where ∇_v denotes the gradient operator with respect to $\bar{v}$. This leads to

$$\bar{v} = -\mathbf{H}^{-1}\left(\bar{x}^k\right)\nabla f\left(\bar{x}^k\right). \tag{12.25}$$

The Newton–Raphson method is thus derived below.

$$f\left(\bar{x}^{k+1}\right) = f\left(\bar{x}^k\right) - \mathbf{H}^{-1}\left(\bar{x}^k\right)\nabla f\left(\bar{x}^k\right). \tag{12.26}$$

Another loose and intuitive way to view the Newton–Raphson method is that its format is similar to the steepest descent method, except that the step size α^k is now chosen as $\mathbf{H}^{-1}$ ($\bar{x}^k$), the inverse of the Hessian matrix evaluated at $\bar{x}^k$.

The idea behind the Newton–Raphson method is that the function being minimized is approximated locally by a quadratic function and this quadratic function is then minimized. It is noted that any function will behave like a quadratic function when it is close to the minimum. Hence, the closer to the minimum, the more efficient the Newton–Raphson method. This is the exact opposite of the steepest descent method, which works more efficiently at the beginning, and less efficiently when close to the minimum. The price paid with the Newton–Raphson method is the extra calculation involved in evaluating the inverse of the Hessian matrix at $\bar{x}^k$.

Convergence Speed — Assume that the second-order sufficient conditions discussed in Section 12.2.2 are satisfied. Furthermore, assume that the initial point $\bar{x}^0$ is sufficiently close to the minimum $\bar{x}^*$. Then it can be shown that the Newton–Raphson method converges with an order of at least two. This indicates that the Newton–Raphson method converges faster than the steepest descent method.

Generalization and Improvements — In Luenberger (1984), a general class of algorithms is defined as

$$\bar{x}^{k+1} = \bar{x}^k - \alpha^k G\nabla f\left(\bar{x}^k\right), \tag{12.27}$$

where G denotes an $n \times n$ matrix, and α^k a positive parameter. Both the steepest descent method and the Newton–Raphson method fall into this framework. It is clear that if G is an $n \times n$ identical matrix $\mathbf{I}$, this general form reduces to the steepest descent method. If $G = \mathbf{H}$ and $\alpha = 1$ then this is the Newton–Raphson method.

Although it descends rapidly near the solution, the Newton–Raphson method may not descend for points far away from the minimum because the quadratic approximation may not be valid there. The introduction of the α^k, which minimizes f, can guarantee the descent of f at the general points. Another improvement is to set $G = [\zeta^k I + H(\bar{x}^k)]^{-1}$ with $\zeta \geq 0$. Obviously, this is a combination of the steepest descent method and the Newton–Raphson method. Two extreme ends are that the steepest method (very large ζ^k) and the Newton–Raphson method ($\zeta^k = 0$). For most cases, the selection of the parameter ζ^k aims at making the G matrix positive definite.

12.2.7 Other Methods

There are other gradient methods such as the Fletcher–Reeves method (also known as the conjugate gradient method) and the Fletcher–Powell–Davidon method (also known as the variable metric method). Readers may refer to Luenberger (1984) or other optimization text.

12.3 THE NETRAVALI–ROBBINS PEL RECURSIVE ALGORITHM

Having had an introduction to some basic nonlinear programming theory, we now turn to the pel recursive technique in displacement estimation from the perspective of the descent methods. Let us take a look at the first pel recursive algorithm, the Netravali–Robbins pel recursive algorithm. It actually estimates displacement vectors using the steepest descent method to minimize the squared DFD. That is,

$$\vec{d}^{k+1} = \vec{d}^{k} - \frac{1}{2}\alpha\nabla_{\vec{d}} DFD^2\left(x, y, \vec{d}^{k}\right), \tag{12.28}$$

where $\nabla_{\vec{d}}DFD^2(x,\ y,\ \vec{d}^k)$ denotes the gradient of DFD^2 with respect to $\vec{d}$ evaluated at $\vec{d}^k$, the displacement vector at the kth iteration, and α is positive. This equation can be further written as

$$\vec{d}^{k+1} = \vec{d}^{k} - \alpha DFD\left(x, y, \vec{d}^{k}\right)\nabla_{\vec{d}} DFD\left(x, y, \vec{d}^{k}\right). \tag{12.29}$$

A a result of Equation 12.1, the above equation leads to

$$\vec{d}^{k+1} = \vec{d}^{k} - \alpha DFD\left(x, y, \vec{d}^{k}\right)\nabla_{x,y} f_{n-1}\left(x - d_x, y - d_y\right), \tag{12.30}$$

where $\nabla_{x,y}$ means a gradient operator with respect to x and y. Netravali and Robbins (1979) assigned a constant of $^1/_{1024}$ to α, i.e., $^1/_{1024}$.

12.3.1 Inclusion of a Neighborhood Area

To make displacement estimation more robust, Netravali and Robbins considered an area for evaluating the DFD^2 in calculating the update term. More precisely, they assume the displacement vector is constant within a small neighborhood Ω of the pixel for which the displacement is being estimated. That is,

$$\vec{d}^{k+1} = \vec{d}^{k} - \frac{1}{2}\alpha\nabla_{\vec{d}} \sum_{i,x,y,\in\Omega} w_i DFD^2\left(x, y, ; \vec{d}^{k}\right), \tag{12.31}$$

where i represents an index for the ith pixel $(x,\ y)$ within Ω, and w_i is the weight for the ith pixel in Ω. All the weights satisfy the following two constraints.

$$\begin{cases} w_i \geq 0 & (12.32) \\ \sum_{i\in\Omega} w_i = 1. & (12.33) \end{cases}$$

This inclusion of a neighborhood area also explains why pel recursive technique is classified into the category of region-matching techniques as we discussed at the beginning of this chapter.

12.3.2 Interpolation

It is noted that interpolation will be necessary when the displacement vector components d_x and d_y are not integer numbers of pixels. A bilinear interpolation technique is used by Netravali and Robbins (1979). For the bilinear interpolation, readers may refer to Chapter 10.

12.3.3 Simplification

To make the proposed algorithm more efficient in computation, Netravali and Robbins also proposed simplified versions of the displacement estimation and interpolation algorithms in their paper.

One simplified version of the Netravali and Robbins algorithm is as follows:

$$\vec{d}^{k+1} = \vec{d}^{k} - \alpha\, sign\left\{DFD\left(x,, \vec{d}^{k}\right)\right\} sign\left\{\nabla_{x,y} f_{n-1}\left(x - d_x, y - d_y\right)\right\}, \tag{12.34}$$

where $sign\{s\} = 0, 1, -1$, depending on $s = 0$, $s > 0$, $s < 0$, respectively, while the sign of a vector quantity is the vector of signs of its components. In this version the update vectors can only assume an angle which is an integer multiple of 45°. As shown in Netravali and Robbins (1979), this version is effective.

12.3.4 Performance

The performance of the Netravali and Robbins algorithm has been evaluated using computer simulation (Netravali and Robbins, 1979). Two video sequences with different amounts and different types of motion are tested. In either case, the proposed pel recursive algorithm displays superior performance over the replenishment algorithm (Mounts, 1969; Haskell, 1979), which was discussed briefly in Chapter 10. The Netravali and Robbins algorithm achieves a bit rate which is 22 to 50% lower than that required by the replenishment technique with the simple frame difference prediction.

12.4 OTHER PEL RECURSIVE ALGORITHMS

The progress and success of the Netravali and Robbins algorithm stimulated great research interests in pel recursive techniques. Many new algorithms have been developed. Some of them are discussed in this section.

12.4.1 The Bergmann Algorithm (1982)

Bergmann modified the Netravali and Robbins algorithm by using the Newton–Raphson method (Bergmann, 1982). In doing so, the following difference between the fundamental framework of the descent methods discussed in Section 12.2 and the minimization problem in displacement estimation discussed in Section 12.3 need to be noticed. That is, the object function $f(\vec{x})$ discussed in Section 12.2 now becomes $DFD^2(x, y, \vec{d})$. The Hessian matrix **H**, consisting of the second-order partial derivatives of the $f(\vec{x})$ with respect to the components of $\vec{x}$ now become the second-order derivatives of DFD^2 with respect to d_x and d_y. Since the vector $\vec{d}$ is a 2-D column vector now, the **H** matrix is hence a 2 × 2 matrix. That is,

$$\mathbf{H} = \begin{bmatrix} \dfrac{\partial^2 DFD^2\left(x, y, \vec{d}\right)}{\partial^2 d_x} & \dfrac{\partial^2 DFD^2\left(x, y, \vec{d}\right)}{\partial d_x \partial d_y} \\ \dfrac{\partial^2 DFD^2\left(x, y, \vec{d}\right)}{\partial d_y \partial d_x} & \dfrac{\partial^2 DFD^2\left(x, y, \vec{d}\right)}{\partial^2 d_y} \end{bmatrix}. \tag{12.35}$$

As expected, the Bergmann algorithm (1982) converges to the minimum faster than the steepest descent method since the Newton–Raphson method converges with an order of at least two.

12.4.2 The Bergmann Algorithm (1984)

Based on the Burkhard and Moll algorithm (Burkhard and Moll, 1979), Bergmann developed an algorithm that is similar to the Newton–Raphson algorithm. The primary difference is that an average of two second-order derivatives is used to replace those in the Hessian matrix. In this sense, it can be considered as a variation of the Newton–Raphson algorithm.

12.4.3 The Cafforio and Rocca Algorithm

Based on their early work (Cafforio and Rocca, 1975), Cafforio and Rocca proposed an algorithm in 1982, which is essentially the steepest descent method. That is, the step size α is defined as follows (Cafforio and Rocca, 1982):

$$\alpha = \frac{1}{\left|\nabla f_{n-1}\left(x-d_x, y-d_y\right)\right|^2 + \eta^2}, \tag{12.36}$$

with $\eta^2 = 100$. The addition of η^2 is intended to avoid the problem that would have occurred in a uniform region where the gradients are very small.

12.4.4 The Walker and Rao Algorithm

Walker and Rao developed an algorithm based on the steepest descent method (Walker and Rao, 1984; Tekalp, 1995), and also with a variable step size. That is,

$$\alpha = \frac{1}{2\left|\nabla f_{n-1}\left(x-d_x, y-d_y\right)\right|^2}, \tag{12.37}$$

where

$$\left|\nabla f_{n-1}\left(x-d_x, y-d_y\right)\right|^2 = \left(\frac{\partial f_{n-1}\left(x-d_x, y-d_y\right)}{\partial d_x}\right)^2 + \left(\frac{\partial f_{n-1}\left(x-d_x, y-d_y\right)}{\partial d_y}\right)^2. \tag{12.38}$$

It is observed that this step size is variable instead of being a constant. Furthermore, this variable step size is reverse proportional to the norm square of the gradient of f_{n-1} $(x - d_x, y - d_y)$ with respect to x, y. That means this type of step size will be small in the edge or rough area, and will be large in the relatively smooth area. These features are desirable.

Although it is quite similar to the Cafforio and Rocca algorithm, the Walker and Rao algorithm differs in the following two aspects. First, the α is selected differently. Second, implementation of the algorithm is different. For instance, instead of putting an η^2 in the denominator of α, the Walker and Rao algorithm uses a logic.

As a result of using the variable step size α, the convergence rate is improved substantially. This implies fast implementation and accurate displacement estimation. It was reported that usually one to three iterations are able to achieve quite satisfactory results in most cases.

Another contribution is that the Walker and Rao algorithm eliminates the need to transmit explicit address information to bring out higher coding efficiency.

12.5 PERFORMANCE COMPARISON

A comprehensive survey of various algorithms using the pel recursive technique can be found in a paper by Musmann, Pirsch, and Grallert (1985). There, two performance features are compared among the algorithms. One is the convergence rate and hence the accuracy of displacement estimation. The other is the stability range. By *stability range*, we mean a range starting from which an algorithm can converge to the minimum of DFD^2, or the true displacement vector.

Compared with the Netravali and Robbins algorithm, those improved algorithms discussed in the previous section do not use a constant step size, thus providing better adaptation to local image

TABLE 12.1
Classification of Several Pel Recursive Algorithms

Algorithms	Category I Steepest Descent Based	Category II Newton–Raphson Based
Netravali and Robbins	Steepest descent	
Bergmann (1982)		Newton–Raphson
Walker and Rao	Variation of steepest descent	
Cafforio and Rocca	Variation of steepest descent	
Bergmann (1984)		Variation of Newton–Raphson

statistics. Consequently, they achieve a better convergence rate and more accurate displacement estimation. According to Bergmann (1984) and Musmann et al. (1985), the Bergmann algorithm (1984) performs best among these various algorithms in terms of convergence rate and accuracy.

According to Musmann et al. (1985), the Newton–Raphson algorithm has a relatively smaller stability range than the other algorithms. This agrees with our discussion in Section 12.2.2. That is, the performance of the Newton–Raphson method improves when it works in the area close to the minimum. The choice of the initial guess, however, is relatively more restricted.

12.6 SUMMARY

The pel recursive technique is one of three major approaches to displacement estimation for motion compensation. It recursively estimates displacement vectors in a pixel-by-pixel fashion. There are three types of recursion: horizontal, vertical, and temporal. Displacement estimation is carried out by minimizing the square of the displaced frame difference (DFD). Therefore, the steepest descent method and the Newton–Raphson method, the two most fundamental methods in optimization, naturally find their application in pel recursive techniques. The pioneering Netravali and Robbins algorithm and several other algorithms such as the Bergmann (1982), the Cafforio and Rocca, the Walker and Rao, and the Bergmann (1984) are discussed in this chapter. They can be classified into one of two categories: the steepest-descent-based algorithms or the Newton–Raphson-based algorithms. Table 12.1 contains a classification of these algorithms.

Note that the DFD can be evaluated within a neighborhood of the pixel for which a displacement vector is being estimated. The displacement vector is assumed constant within this neighborhood. This makes the displacement estimation more robust against various noises.

Compared with the replenishment technique with simple frame difference prediction (the first real interframe coding algorithm), the Netravali and Robbins algorithm (the first pel recursive technique) achieves much higher coding efficiency. Specifically, a 22 to 50% savings in bit rate has been reported for some computer simulations. Several new pel recursive algorithms have made further improvements in terms of the convergence rate and the estimation accuracy through replacement of the fixed step size utilized in the Netravali and Robbins algorithm, which make these algorithms more adaptive to the local statistics in image frames.

12.7 EXERCISES

12-1. What is the definition of the displaced frame difference? Justify Equation 12.1.

12-2. Why does the inclusion of a neighborhood area make the pel recursive algorithm more robust against noise?

12-3. Compare the performance of the steepest descent method with that of the Newton–Raphson method.

12-4. Explain the function of η^2 in the Cafforio and Rocca algorithm.

12-5. What is the advantage you expect to have from the Walker and Rao algorithm?

12-6. What is the difference between the Bergmann algorithm (1982) and the Bergmann algorithm (1984)?

12-7. Why does the Newton–Raphson method have a smaller stability range?

REFERENCES

Bergmann, H. C. Displacement estimation based on the correlation of image segments, *IEEE Proceedings of International Conference on Electronic Image Processing*, 215-219, York, U.K., July 1982.

Bergmann, H. C. Ein Schnell Konvergierendes Displacement-Schätzverfahrenfür die Interpolation von Fernsehbildsequenzen, Ph.D. dissertation, Technical University of Hannover, Hannover, Germany, February 1984.

Biemond, J., L. Looijenga, D. E. Boekee, and R. H. J. M. Plompen, A pel recursive Wiener-based displacement estimation algorithm, *Signal Processing*, 13, 399-412, December 1987.

Burkhard, H. and H. Moll, A modified Newton–Raphson search for the model-adaptive identification of delays, in *Identification and System Parameter Identification*, R. Isermann, Ed., Pergamon Press, New York, 1979, 1279-1286.

Cafforio, C. and F. Rocca, The differential method for image motion estimation, in *Image Sequence Processing and Dynamic Scene Analysis*, T. S. Huang, Ed., Berlin, Germany: Springer-Verlag, New York, 1983, 104-124.

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

Luenberger, D. G. *Linear and Nonlinear Programming*, Addison Wesley, Reading, MA, 1984.

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.

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

Tekalp, A. M. *Digital Video Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1995.

Walker, D. R. and K. R. Rao, Improved pel-recursive motion compensation, *IEEE Trans. Commun.*, COM-32, 1128-1134, 1984.

13 Optical Flow

As mentioned in Chapter 10, optical flow is one of three major techniques that can be used to estimate displacement vectors from successive image frames. As opposed to the other two displacement estimation techniques discussed in Chapters 11 and 12, block matching and pel recursive method, however, the optical flow technique was developed primarily for 3-D motion estimation in the computer vision community. Although it provides a relatively more accurate displacement estimation than the other two techniques, as we shall see in this and the next chapter, optical flow has not yet found wide applications for motion-compensated video coding. This is mainly due to the fact that there are a large number of motion vectors (one vector per pixel) involved, hence, the more side information that needs to be encoded and transmitted. As emphasized in Chapter 11, we should not forget the ultimate goal in motion-compensated video coding: to encode video data with a *total* bit rate as low as possible, while maintaining a satisfactory quality of reconstructed video frames at the receiving end. If the extra bits required for encoding a large amount of optical flow vectors counterbalance the bits saved in encoding the prediction error (as a result of more accurate motion estimation), then the usage of optical flow in motion-compensated coding is not worthwhile. Besides, more computation is required in optical flow determination. These factors have prevented optical flow from being practically utilized in motion-compensated video coding. With the continued advance in technologies, however, we believe this problem may be resolved in the near future. In fact, an initial, successful attempt has been made (Shi et al., 1998).

On the other hand, in theory, the optical flow technique is of great importance in understanding the fundamental issues in 2-D motion determination, such as the aperture problem, the conservation and neighborhood constraints, and the distinction and relationship between 2-D motion and 2-D apparent motion.

In this chapter we focus on the optical flow technique. In Section 13.1, as stated above, some fundamental issues associated with optical flow are addressed. Section 13.2 discusses the differential method. The correlation method is covered in Section 13.3. In Section 13.4, a multiple attributes approach is presented. Some performance comparisons between various techniques are included in Sections 13.3 and 13.4. A summary is given in Section 13.5.

13.1 FUNDAMENTALS

Optical flow is referred to as the 2-D distribution of apparent velocities of movement of intensity patterns in an image plane (Horn and Schunck, 1981). In other words, an optical flow field consists of a dense velocity field with one velocity vector for each pixel in the image plane. If we know the time interval between two consecutive images, which is usually the case, then velocity vectors and displacement vectors can be converted from one to another. In this sense, optical flow is one of the techniques used for displacement estimation.

13.1.1 2-D Motion and Optical Flow

In the above definition, it is noted that the word *apparent* is used and nothing about 3-D motion in the scene is stated. The implication behind this observation is discussed in this subsection. We start with the definition of 2-D motion. 2-D motion is referred to as motion in a 2-D image plane caused by 3-D motion in the scene. That is, 2-D motion is the projection (commonly perspective projection) of 3-D motion in the scene onto the 2-D image plane. This can be illustrated by using

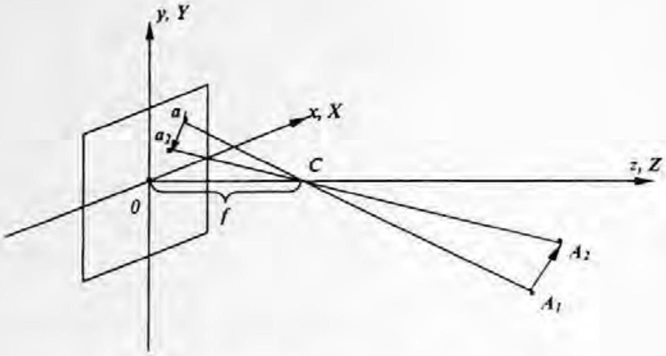


FIGURE 13.1 2-D motion vs. 3-D motion.

a very simple example, shown in Figure 13.1. There the world coordinate system *O-XYZ* and the camera coordinate systems *o-xyz* are aligned. The point *C* is the optical center of the camera. A point A_1 moves to A_2, while its perspective projection moves correspondingly from a_1 to a_2. We then see that a 2-D motion (from a_1 to a_2) in image plane is invoked by a 3-D motion (from A_1 to A_2) in 3-D space. By a 2-D motion field, or sometimes image flow, we mean a dense 2-D motion field: One velocity vector for each pixel in the image plane.

Optical flow, according to its definition, is caused by movement of intensity patterns in an image plane. Therefore 2-D motion (field) and optical flow (field) are generally different. To support this conclusion, let us consider the following two examples. One is given by Horn and Schunck (1981). Imagine a uniform sphere rotating with a constant speed in the scene. Assume the luminance and all other conditions do not change at all when pictures are taken. Then, there is no change in brightness patterns in the images. According to the definition of optical flow, the optical flow is zero, whereas the 2-D motion field is obviously not zero. At the other extreme, consider a stationary scene; all objects in 3-D world space are still. If illuminance changes when pictures are taken in such a way that there is movement of intensity patterns in image planes, as a consequence, optical flow may be nonzero. This confirms a statement made by Singh (1991): the scene does not have to be in motion relative to the image for the optical flow field to be nonzero. It can be shown that the 2-D motion field and the optical flow field are equal under certain conditions. Understanding the difference between the two quantities and the conditions under which they are equal is important.

This understanding can provide us with some sort of guide to evaluate the reliability of estimating 3-D motion from optical flow. This is because, in practice, time-varying image sequences are only what we have at hand. The task in computer vision is to interpret 3-D motion from time-varying sequences. Therefore, we can only work with optical flow in estimating 3-D motion. Since the main focus of this book is on image and video coding, we do not cover these equality conditions here. Interested readers may refer to Singh (1991). In motion-compensated video coding, it is likewise true that the image frames and video data are only what we have at hand. We also, therefore, have to work with optical flow. Our attention is thus turned to optical flow determination and its usage in video data compression.

13.1.2 Aperture Problem

The aperture problem is an important issue, originating in optics. Since it is inherent in the local estimation of optical flow, we address this issue in this subsection. In optics, apertures are openings in flat screens (Bracewell, 1995). Therefore, apertures can have various shapes, such as circular, semicircular, and rectangular. Examples of apertures include a thin slit or array of slits in a screen. A circular aperture, a round hole made on the shutter of a window, was used by Newton to study the composition of sunlight. It is also well known that the circular aperture is of special interest in studying the diffraction pattern (Sears et al., 1986).

Roughly speaking, the aperture problem in motion analysis refers to the problem that occurs when viewing motion via an aperture, i.e., a small opening in a flat screen. Marr (1982) states that when a straight moving edge is observed through an aperture, only the component of motion orthogonal to the edge can be measured. Let us examine some simple examples depicted in Figure 13.2. In Figure 13.2(a), a large rectangular *ABCD* is located in the *XOZ* plane. A rectangular screen *EFGH* with a circular aperture is perpendicular to the *OY* axis. Figure 13.2(b) and (c) show, respectively, what is observed through the aperture when the rectangular *ABCD* is moving along the positive *X* and *Z* directions with a uniform speed. Since the circular opening is small and the line *AB* is very long, no motion will be observed in Figure 13.2(b). Obviously, in Figure 13.2(c) the upward movement can be observed clearly. In Figure 13.2(d), the upright corner of the rectangle *ABCD*, angle *B*, appears. At this time the translation along any direction in the *XOZ* plane can be observed clearly. The phenomena observed in this example demonstrate that it is sometimes impossible to estimate motion of a pixel by only observing a small neighborhood surrounding it. The only motion that can be estimated from observing a small neighborhood is the motion orthogonal to the underlying moving contour. In Figure 13.2(b), there is no motion orthogonal to the moving contour *AB*; the motion is aligned with the moving contour *AB*, which cannot be observed through the aperture. Therefore, no motion can be observed through the aperture. In Figure 13.2(c), the observed motion is upward, which is perpendicular to the horizontal moving contour *AB*. In Figure 13.2(d), any translation in the *XOZ* plane can be decomposed into horizontal and vertical components. Either of these two components is orthogonal to one of the two moving contours: *AB* or *BC*.

A more accurate statement on the aperture problem needs a definition of the so-called normal optical flow. The normal optical flow refers to the component of optical flow along the direction pointed by the local intensity gradient. Now we can make a more accurate statement: the only motion in an image plane that can be determined is the normal optical flow.

In general, the aperture problem becomes severe in image regions where strong intensity gradients exist, such as at the edges. In image regions with strong higher-order intensity variations, such as corners or textured areas, the true motion can be estimated. Singh (1991) provides a more elegant discussion on the aperture problem, in which he argues that the aperture problem should be considered as a continuous problem (it always exists, but in varying degrees of acuteness) instead of a binary problem (either it exists or it does not).

13.1.3 Ill-Posed Inverse Problem

Motion estimation from image sequences, including optical flow estimation, belongs in the category of inverse problems. This is because we want to infer motion from given 2-D images, which is the perspective projection of 3-D motion. According to Hadamard (Bertero et al., 1988), a mathematical problem is well posed if it possesses the following three characteristics:

1. Existence. That is, the solution exists.
2. Uniqueness. That is, the solution is unique.
3. Continuity. That is, when the error in the data tends toward zero, then the induced error in the solution tends toward zero as well.

Inverse problems usually are not well posed in that the solution may not exist. In the example discussed in Section 13.1.1, i.e., a uniform sphere rotated with illuminance fixed, the solution to motion estimation does not exist since no motion can be inferred from given images. The aperture problem discussed in Section 13.1.2 is the case in which the solution to the motion may not be unique. Let us take a look at Figure 13.2(b). From the given picture, one cannot tell whether the straight line *AB* is static, or is moving horizontally. If it is moving horizontally, one cannot tell the moving speed. In other words, infinitely many solutions exist for the case. In optical flow determination, we will

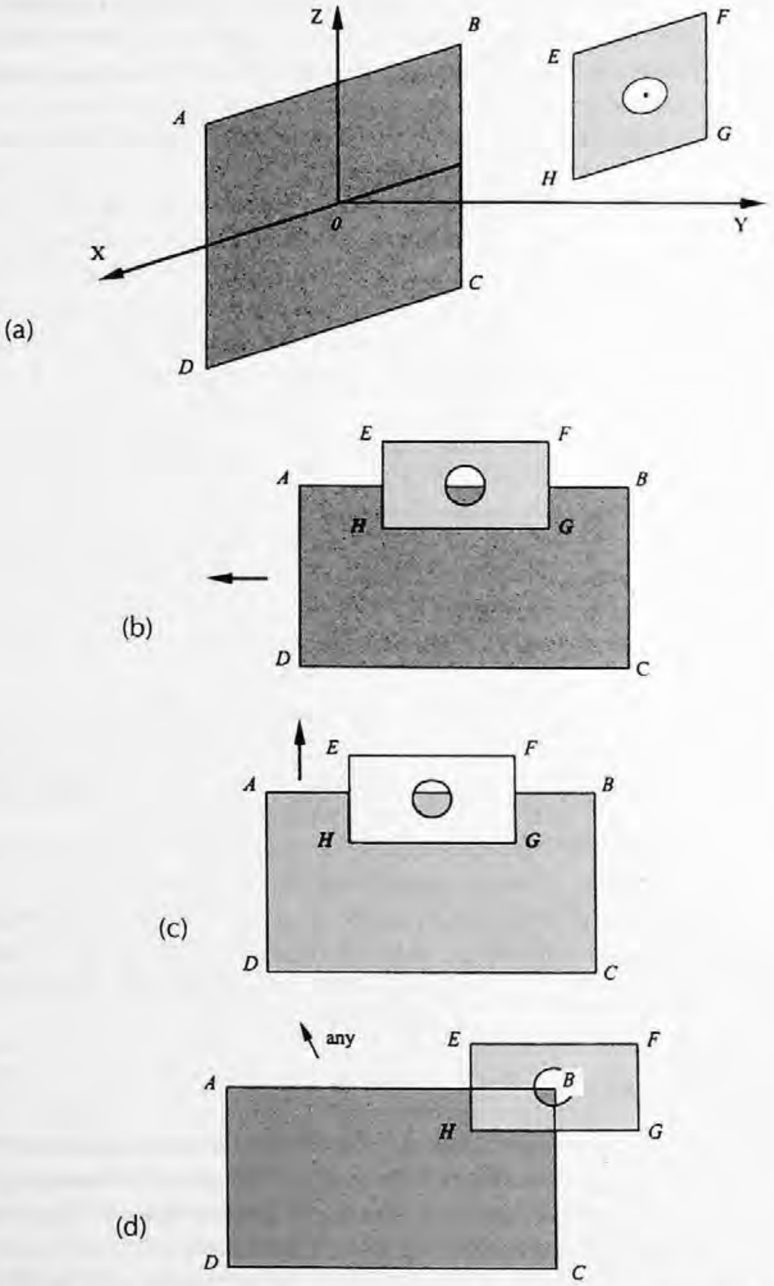


FIGURE 13.2 (a) Aperture problem: A large rectangle ABCD is located in the *XOZ* plane. A rectangular screen *EFGH* with a circular aperture is perpendicular to the *OY* axis. (b) Aperture problem: No motion can be observed through the circular aperture when the rectangular *ABCD* is moving along the positive *X* direction. (c) Aperture problem: The motion can be observed through the circular aperture when the *ABCD* is moving along the positive *Z* direction. (d) Aperture problem: The translation of *ABCD* along any direction in the *XOZ* plane can be observed through the circular aperture when the upright corner of the rectangle *ABCD*, angle *B*, appears in the aperture.

see that computations are noise sensitive. That is, even a small error in the data can produce an extremely large error in the solution. Hence, we see that the motion estimation from image sequences suffers from all three aspects just mentioned: nonexistence, nonuniqueness, and discontinuity. The last term is also referred to as the instability of the solution.

It is pointed out by Bertero et al. (1988) that all the low-level processing (also known as early vision) in computational vision are inverse problems and are often ill posed. Examples in low-level processing include motion recovery, computation of optical flow, edge detection, structure from stereo, structure from motion, structure from texture, shape from shading, and so on. Fortunately, the problem with early vision is mildly ill posed in general. By *mildly*, we mean that a reduction of errors in the data can significantly improve the solution.

Since the early 1960s, the demand for accurate approximates and stable solutions in areas such as optics, radioastronomy, microscopy, and medical imaging has stimulated great research efforts in inverse problems, resulting in a unified theory: the regularization theory of ill-posed problems (Tikhonov and Arsenin, 1977). In the discussion of optical flow methods, we shall see that some regularization techniques have been posed and have improved accuracy in flow determination. More-advanced algorithms continue to come.

13.1.4 Classification of Optical Flow Techniques

Optical flow in image sequences provides important information regarding both motion and structure, and it is useful in such diverse fields as robot vision, autonomous navigation, and video coding. Although this subject has been studied for more than a decade, reducing the error in the flow estimation remains a difficult problem. A comprehensive review and a comparison of the accuracy of various optical flow techniques have recently been made (Barron et al., 1994). So far, most of the techniques in the optical flow computations use one of the following basic approaches:

- Gradient-based (Horn and Schunck, 1981; Lucas and Kanade, 1981; Nagel and Enkelman, 1986; Uras et al., 1988; Szeliski et al., 1995; Black and Anandan, 1996),
- Correlation-based (Anandan, 1989; Singh, 1992; Pan et al., 1998),
- Spatiotemporal energy-based (Adelson and Bergen, 1985; Heeger, 1988; Bigun et al., 1991),
- Phase-based (Waxman et al., 1988; Fleet and Jepson, 1990).

Besides these deterministic approaches, there is the stochastic approach to optical flow computation (Konrad and Dubois, 1992). In this chapter we focus our discussion of optical flow on the gradient-based and correlation-based techniques because of their frequent applications in practice and fundamental importance in theory. We also discuss multiple attribute techniques in optical flow determination. The other two approaches will be briefly touched upon when we discuss new techniques in motion estimation in the next chapter.

13.2 GRADIENT-BASED APPROACH

It is noted that before the methods of optical flow determination were actually developed, optical flow had been discussed and exploited for motion and structure recovery from image sequences in computer vision for years. That is, the optical flow field was assumed to be available in the study of motion recovery. The first type of methods in optical flow determination is referred to as gradient-based techniques. This is because the spatial and temporal partial derivatives of intensity function are utilized in these techniques. In this section, we present the Horn and Schunck algorithm. It is regarded as the most prominent representative of this category. After the basic concepts are presented, some other methods in this category are briefly discussed.

13.2.1 The Horn and Schunck Method

We shall begin with a very general framework (Shi et al., 1994) to derive a brightness time-invariance equation. We then introduce the Horn and Schunck method.

13.2.1.1 Brightness Invariance Equation

As stated in Chapter 10, the imaging space can be represented by

$$f(x, y, t, \bar{s}), \tag{13.1}$$

where $\bar{s}$ indicates the sensor's position in 3-D world space, i.e., the coordinates of the sensor center and the orientation of the optical axis of the sensor. The $\bar{s}$ is a 5-D vector. That is, $\bar{s}$ where ($\tilde{x}$, $\tilde{y}$, $\tilde{z}$, β, γ), where $\tilde{x}$, $\tilde{y}$, and $\tilde{z}$ represent the coordinate of the optical center of the sensor in 3-D world space; and β and γ represent the orientation of the optical axis of the sensor in 3-D world space, the Euler angles, pan and tilt, respectively.

With this very general notion, each picture, which is 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 become a proper subset of the imaging space.

Assume now a world point P in 3-D space that is perspectively 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,

$$f = f\left(x_P(t, \bar{s}), y_P(t, \bar{s}), t, \bar{s}\right). \tag{13.2}$$

If the optical radiation of the world point P is invariant with respect to the time interval from t_1 to t_2, we then have

$$f\left(x_P(t_1, \bar{s}_1), y_P(t_1, \bar{s}_1), t_1, \bar{s}_1\right) = f\left(x_P(t_2, \bar{s}_1), y_P(t_2, \bar{s}_1), t_2, \bar{s}_1\right). \tag{13.3}$$

This is the brightness time-invariance equation.

At a specific moment t_1, if the optical radiation of P is isotropical we then get

$$f\left(x_P(t_1, \bar{s}_1), y_P(t_1, \bar{s}_1), t_1, \bar{s}_1\right) = f\left(x_P(t_1, \bar{s}_2), y_P(t_1, \bar{s}_2), t_1, \bar{s}_2\right). \tag{13.4}$$

This is the brightness space-invariance equation.

If both conditions are satisfied, we get the brightness time-and-space-invariance equation, i.e.,

$$f\left(x_P(t_1, \bar{s}_1), y_P(t_1, \bar{s}_1), t_1, \bar{s}_1\right) = f\left(x_P(t_2, \bar{s}_2), y_P(t_2, \bar{s}_2), t_2, \bar{s}_2\right). \tag{13.5}$$

Consider two brightness functions $f(x(t, \bar{s}), y(t, \bar{s}), t, \bar{s})$ and $f(x(t + \Delta t, \bar{s} + \Delta\bar{s}), y(t + \Delta t, \bar{s} + \Delta\bar{s}), t + \Delta t, \bar{s} + \Delta\bar{s})$ in which the variation in time, Δt, and the variation in the spatial position of the sensor, $\Delta\bar{s}$, are very small. Due to the time-and-space-invariance of brightness, we can get

$$f\left(x(t, \bar{s}), y(t, \bar{s}), t, \bar{s}\right) = f\left(x(t + \Delta t, \bar{s} + \Delta\bar{s}), y(t + \Delta t, \bar{s} + \Delta s), t + \Delta t, \bar{s} + \Delta\bar{s}\right). \tag{13.6}$$

The expansion of the right-hand side of the above equation in the Taylor series at $(t, \bar{s})$ and the use of Equation 13.5 lead to

$$\left(\frac{\partial f}{\partial x}u + \frac{\partial f}{\partial y}v + \frac{\partial f}{\partial t}\right)\Delta t + \left(\frac{\partial f}{\partial x}u^{\bar{s}} + \frac{\partial f}{\partial y}v^{\bar{s}} + \frac{\partial f}{\partial \bar{s}}\right)\Delta\bar{s} + \varepsilon = 0, \tag{13.7}$$

where

$$u = \frac{\partial x}{\partial t}, \quad v \overset{\ddot{\mathrm{A}}}{=} \frac{\partial y}{\partial t}, \quad u^{\bar{s}} \overset{\ddot{\mathrm{A}}}{=} \frac{\partial x}{\partial \bar{s}}, \quad u^{\bar{s}} \overset{\ddot{\mathrm{A}}}{=} \frac{\partial y}{\partial \bar{s}}.$$

If $\Delta\bar{s} = 0$, i.e., the sensor is static in a fixed spatial position (in other words, both the coordinate of the optical center of the sensor and its optical axis direction remain unchanged), dividing both sides of the equation by Δt and evaluating the limit as $\Delta t \to 0$ degenerate Equation 13.7 into

$$\frac{\partial f}{\partial x}u + \frac{\partial f}{\partial y}v + \frac{\partial f}{\partial t} = 0. \tag{13.8}$$

If $\Delta t = 0$, both its sides are divided by $\Delta\bar{s}$, and $\Delta\bar{s} \to 0$ is examined. Equation 13.7 then reduces to

$$\frac{\partial f}{\partial x}u^{\bar{s}} + \frac{\partial f}{\partial y}v^{\bar{s}} + \frac{\partial f}{\partial \bar{s}} = 0. \tag{13.9}$$

When $\Delta t = 0$, i.e., at a specific time moment, the images generated with sensors at different spatial positions can be viewed as a spatial sequence of images. Equation 13.9 is, then, the equation for the spatial sequence of images.

For the sake of brevity, we will focus on the gradient-based approach to optical flow determination with respect to temporal image sequences. That is, in the rest of this section we will address only Equation 13.8. It is noted that the derivation can be extended to spatial image sequences. The optical flow technique for spatial image sequences is useful in stereo image data compression. It plays an important role in motion and structure recovery. Interested readers are referred to Shi et al. (1994) and Shu and Shi (1993).

13.2.1.2 Smoothness Constraint

A careful examination of Equation 13.8 reveals that we have two unknowns: *u* and *v*, i.e., the horizontal and vertical components of an optical flow vector at a three-tuple (*x*, *y*, *t*), but only one equation to relate them. This once again demonstrates the ill-posed nature of optical flow determination. This also indicates that there is no way to compute optical flow by considering a single point of the brightness pattern moving independently. As stated in Section 13.1.3, some regularization measure — here an extra constraint — must be taken to overcome the difficulty.

A most popularly used constraint was proposed by Horn and Schunck and is referred to as the smoothness constraint. As the name implies, it constrains flow vectors to vary from one to another smoothly. Clearly, this is true for points in the brightness pattern most of the time, particularly for points belonging to the same object. It may be violated, however, along moving boundaries. Mathematically, the smoothness constraint is imposed in optical flow determination by minimizing the square of the magnitude of the gradient of the optical flow vectors:

$$\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2. \tag{13.10}$$

It can be easily verified that the smoother the flow vector field, the smaller these quantities. Actually, the square of the magnitude of the gradient of intensity function with respect to the spatial coordinates, summed over a whole image or an image region, has been used as a smoothness measure of the image or the image region in the digital image processing literature (Gonzalez and Woods, 1992).

13.2.1.3 Minimization

Optical flow determination can then be converted into a minimization problem.

The square of the left-hand side of Equation 13.8, which can be derived from the brightness time-invariance equation, represents one type of error. It may be caused by quantization noise or other noises and can be written as

$$\varepsilon_b^2 = \left(\frac{\partial f}{\partial x} u + \frac{\partial f}{\partial y} v + \frac{\partial f}{\partial t} \right)^2 . \tag{13.11}$$

The smoothness measure expressed in Equation 13.10 denotes another type of error, which is

$$\varepsilon_s^2 = \left(\frac{\partial u}{\partial x} \right)^2 + \left(\frac{\partial u}{\partial y} \right)^2 + \left(\frac{\partial v}{\partial x} \right)^2 + \left(\frac{\partial v}{\partial y} \right)^2 . \tag{13.12}$$

The total error to be minimized is

$$\begin{aligned} \varepsilon^2 &= \sum_x \sum_y \varepsilon_b^2 + \alpha^2 \varepsilon_s^2 \\ &= \sum_x \sum_y \left(\frac{\partial f}{\partial x} u + \frac{\partial f}{\partial y} v + \frac{\partial f}{\partial t} \right)^2 + \alpha^2 \left[\left(\frac{\partial u}{\partial x} \right)^2 + \left(\frac{\partial u}{\partial y} \right)^2 + \left(\frac{\partial v}{\partial x} \right)^2 + \left(\frac{\partial v}{\partial y} \right)^2 \right], \end{aligned} \tag{13.13}$$

where α is a weight between these two types of errors. The optical flow quantities u and v can be found by minimizing the total error. Using the calculus of variation, Horn and Schunck derived the following pair of equations for two unknown u and v at each pixel in the image.

$$\begin{cases} f_x^2 u + f_x f_y v = \alpha^2 \nabla^2 u - f_x f_t \\ f_x f_y u + f_y^2 v = \alpha^2 \nabla^2 v - f_y f_t \end{cases} , \tag{13.14}$$

where

$$f_x = \frac{\partial f}{\partial x}, \quad f_y = \frac{\partial f}{\partial y}, \quad f_t = \frac{\partial f}{\partial t};$$

∇^2 denotes the Laplacian operator. The Laplacian operator of u and v are defined below.

$$\begin{aligned} \nabla^2 u &= \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \\ \nabla^2 v &= \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} . \end{aligned} \tag{13.15}$$

13.2.1.4 Iterative Algorithm

Instead of using the classical algebraic method to solve the pair of equations for u and v, Horn and Schunck adopted the Gaussian Seidel (Ralston and Rabinowitz, 1978) method to have the following iterative procedure:

$$u^{k+1} = \bar{u}^k - \frac{f_x\left[f_x\bar{u}^k + f_y\bar{v}^k + f_t\right]}{\alpha^2 + f_x^2 + f_y^2}$$
$$v^{k+1} = \bar{v}^k - \frac{f_y\left[f_x\bar{u}^k + f_y\bar{v}^k + f_t\right]}{\alpha^2 + f_x^2 + f_y^2}, \tag{13.16}$$

where the superscripts k and $k + 1$ are indexes of iteration and $\bar{u}$, $\bar{v}$ are the local averages of u and v, respectively.

Horn and Schunck define $\bar{u}$, $\bar{v}$ as follows:

$$\begin{aligned}\bar{u} &= \frac{1}{6}\{u(x,y+1) + u(x,y-1) + u(x+1,y) + u(x-1,y)\}\\ &+ \frac{1}{12}\{u(x-1,y-1) + u(x-1,y+1) + u(x+1,y-1) + u(x+1,y+1)\}\\ \bar{v} &= \frac{1}{6}\{v(x,y+1) + v(x,y-1) + v(x+1,y) + v(x-1,y)\}\\ &+ \frac{1}{12}\{v(x-1,y-1) + v(x-1,y+1) + v(x+1,y-1) + v(x+1,y+1)\}.\end{aligned} \tag{13.17}$$

The estimation of the partial derivatives of intensity function and the Laplacian of flow vectors need to be addressed. Horn and Schunck considered a 2 × 2 × 2 spatiotemporal neighborhood, shown in Figure 13.3, for estimation of partial derivatives f_x, f_y, and f_t. Note that replacing the first-order differentiation by the first-order difference is a common practice in managing digital images. The arithmetic average can remove the noise effect, thus making the obtained first-order differences less sensitive to various noises.

The Laplacian of u and v are approximated by

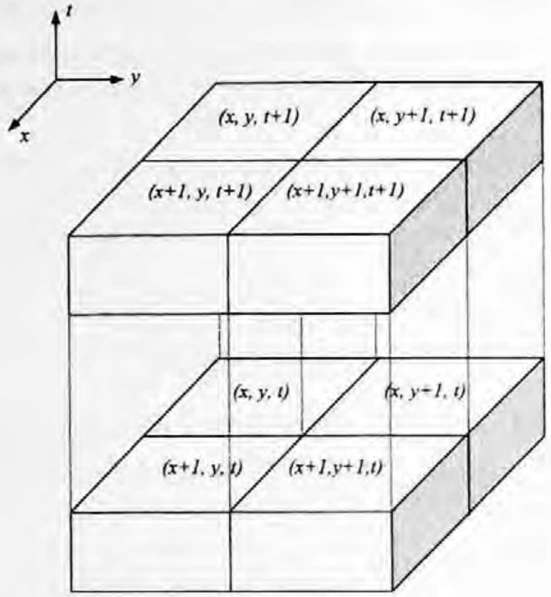
$$\nabla^2 u = \bar{u}(x,y) - u(x,y)$$
$$\nabla^2 v = \bar{v}(x,y) - v(x,y). \tag{13.18}$$

Equivalently, the Laplacian of u and v, $\nabla^2(u)$ and $\nabla^2(v)$, can be obtained by applying a 3 × 3 window operator, shown in Figure 13.4, to each point in the u and v planes, respectively.

Similar to the pel recursive technique discussed in the previous chapter, there are two different ways to iterate. One way is to iterate at a pixel until a solution is steady. Another way is to iterate only once for each pixel. In the latter case, a good initial flow vector is required and is usually derived from the previous pixel.

13.2.2 Modified Horn and Schunck Method

Observing that the first-order difference is used to approximate the first-order differentiation in Horn and Schunck's original algorithm, and regarding this as a relatively crude form and a source



$$f_x = \frac{1}{4}\{[f(x+1,y,t)-f(x,y,t)]+[f(x+1,y,t+1)-f(x,y,t+1)]$$

$$+[f(x+1,y+1,t)-f(x,y,t)]+[f(x+1,y+1,t+1)-f(x,y+1,t+1)]\}$$

$$f_y = \frac{1}{4}\{[f(x,y+1,t)-f(x,y,t)]+[f(x+1,y+1,t)-f(x+1,y,t)]$$

$$+[f(x,y+1,t+1)-f(x,y,t+1)]+[f(x+1,y+1,t+1)-f(x+1,y,t+1)]\}$$

$$f_x = \frac{1}{4}\{[f(x,y,t+1)-f(x,y,t)]+[f(x+1,y,t+1)-f(x+1,y,t)]$$

$$+[f(x,y+1,t+1)-f(x,y+1,t)]+[f(x+1,y+1,t+1)-f(x+1,y+1,t)]\}$$

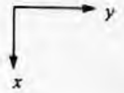
FIGURE 13.3 Estimation of f_x, f_y, and f_t.

of error, Barron, Fleet, and Beauchemin developed a modified version of the Horn and Schunck method (Barron et al., 1994).

It features a spatiotemporal presmoothing and a more-advanced approximation of differentiation. Specifically, it uses a Gaussian filter as a spatiotemporal prefilter. By the term *Gaussian filter*, we mean a low-pass filter with a mask shaped similar to that of the Gaussian probability density function. This is similar to what was utilized in the formulation of the Gaussian pyramid, which was discussed in Chapter 11. The term *spatiotemporal* means that the Gaussian filter is used for low-pass filtering in both spatial and temporal domains.

With respect to the more-advanced approximation of differentiation, a four-point central difference operator is used, which has a mask, shown in Figure 13.5.

As we will see later in this chapter, this modified Horn and Schunck algorithm has achieved better performance than the original one as a result of the two above-mentioned measures. This success indicates that a reduction of noise in image (data) leads to a significant reduction of noise in optical flow (solution). This example supports the statement we mentioned earlier that the ill-posed problem in low-level computational vision is mildly ill posed.



(x-1, y-1)	(x-1, y)	(x-1, y+1)
(x, y-1)	(x, y)	(x, y+1)
(x+1, y-1)	(x+1, y)	(x+1, y+1)

$\frac{1}{12}$	$\frac{1}{6}$	$\frac{1}{12}$
$\frac{1}{6}$	-1	$\frac{1}{6}$
$\frac{1}{12}$	$\frac{1}{6}$	$\frac{1}{12}$

$$\begin{aligned}\nabla^2 u \approx\ & \frac{1}{6}\left[u(x-1,y)+u(x,y-1)+u(x,y+1)+u(x+1,y)\right]\\ & +\frac{1}{12}\left[u(x-1,y-1)+u(x-1,y+1)+u(x+1,y-1)+u(x+1,y+1)\right]\\ & -u(x,y)\end{aligned}$$

$$\begin{aligned}\nabla^2 v \approx\ & \frac{1}{6}\left[v(x-1,y)+v(x,y-1)+v(x,y+1)+v(x+1,y)\right]\\ & +\frac{1}{12}\left[v(x-1,y-1)+v(x-1,y+1)+v(x+1,y-1)+v(x+1,y+1)\right]\\ & -v(x,y)\end{aligned}$$

FIGURE 13.4 A 3 × 3 window operation for estimation of the Laplacian of flow vector.

$-\frac{1}{12}$	$\frac{8}{12}$	0	$-\frac{8}{12}$	$\frac{1}{12}$

FIGURE 13.5 Four-point central difference operator mask.

13.2.3 The Lucas and Kanade Method

Lucas and Kanade assume a flow vector is constant within a small neighborhood of a pixel, denoted by Ω. Then they form a weighted object function as follows.

$$\sum_{(x,y)\in\Omega} w^2(x,y)\left[\frac{\partial f(x,y,t)}{\partial x}u+\frac{\partial f(x,y,t)}{\partial v}v+\frac{\partial f(x,y,t)}{\partial t}\right]^2, \tag{13.19}$$

where $w(x, y)$ is a window function, which gives more weight to the central portion than the surrounding portion of the neighborhood Ω.

The flow determination thus becomes a problem of a least-square fit of the brightness invariance constraint. We observe that the smoothness constraint has been implied in Equation 13.19, where the flow vector is assumed to be constant within Ω.

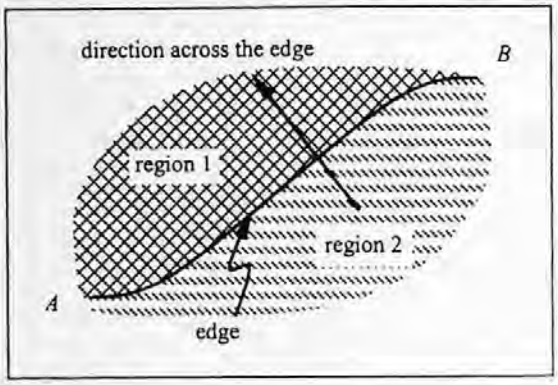


FIGURE 13.6 Oriented-smoothness constraint.

13.2.4 The Nagel Method

Nagel first used the second-order derivatives in optical flow determination in the very early days (Nagel, 1983). Since the brightness function $f(x, y, t, \vec{s})$ is a real-valued function of multiple variables (or a vector of variables), the Hessian matrix, discussed in Chapter 12, is used for the second-order derivatives.

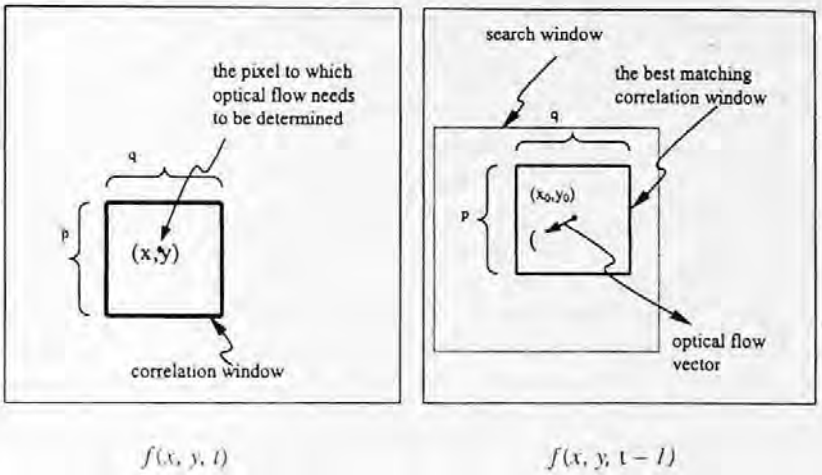
An oriented-smoothness constraint was developed by Nagel that prohibits imposition of the smoothness constraint across edges, as illustrated in Figure 13.6. In the figure, an edge *AB* separates two different moving regions: region 1 and region 2. The smoothness constraint is imposed in these regions separately. That is, no smoothness constraint is imposed across the edge. Obviously, it would be a disaster if we smoothed the flow vectors across the edge. As a result, this reasonable treatment effectively improves the accuracy of optical flow estimation (Nagel, 1989).

13.2.5 The Uras, Girosi, Verri, and Torre Method

The Uras, Girosi, Verri, and Torre method is another method that uses second-order derivatives. Based on a local procedure, it performs quite well (Uras et al., 1988).

13.3 CORRELATION-BASED APPROACH

The correlation-based approach to optical flow determination is similar to block matching, covered in Chapter 11. As may be recalled, the conventional block-matching technique partitions an image into nonoverlapped, fixed-size, rectangular blocks. Then, for each block, the best matching in the previous image frame is found. In doing so, a search window is opened in the previous frame according to some *a priori* knowledge: the time interval between the two frames and the maximum possible moving velocity of objects in frames. Centered on each of the candidate pixels in the search window, a rectangle correlation window of the same size as the original block is opened. The best-matched block in the search window is chosen such that either the similarity measure is maximized or the dissimilarity measure is minimized. The relative spatial position between these two blocks (the original block in the current frame and the best-matched one in the previous frame) gives a translational motion vector to the original block. In the correlation-based approach to optical flow computation, the mechanism is very similar to that in conventional block matching. The only difference is that for each pixel in an image, we open a rectangle correlation window centered on this pixel for which an optical flow vector needs to be determined. It is for this correlation window that we find the best match in the search window in its temporal neighboring image frame. This is shown in Figure 13.7. A comparison between Figures 13.7 and 11.1 can convince us about the



$f(x, y, t)$ $f(x, y, t-1)$

FIGURE 13.7 Correlation-based approach to optical flow determination.

above observation. In this section, we first briefly discuss Anandan's method, which is pioneer work in this category. Then Singh's method is described. His unified view of optical flow computation is introduced. We then present a correlation-feedback method by Pan, Shi, and Shu, which uses the feedback technique in flow calculation.

13.3.1 The Anandan Method

As mentioned in Chapter 11, the sum of squared difference (SSD) is used as a dissimilarity measure in (Anandan, 1987). It is essentially a simplified version of the well-known mean square error (MSE). Due to its simplicity, it is used in the methods developed by Singh (1992), and Pan, Shi, and Shu (1998).

In the Anandan method (Anandan, 1989), a pyramid structure is formed, and it can be used for an efficient coarse-fine search. This is very similar to the multiresolution block-matching techniques discussed in Chapter 11. In the higher levels (with lower resolution) of the pyramid, a full search can be performed without a substantial increase in computation. The estimated velocity (or displacement) vector can be propagated to the lower levels (with higher resolution) for further refinement. As a result, a relatively large motion vector can be estimated with a certain degree of accuracy.

Instead of the Gaussian pyramid discussed in Chapter 11, however, a Laplacian pyramid is used here. To understand the Laplacian pyramid, let us take a look at Figure 13.8(a). There two consecutive levels are shown in a Gaussian pyramid structure: level k, denoted by $f^k(x, y)$, and level $k + 1$, $f^{k+1}(x, y)$. Figure 13.8(b) shows how level $k + 1$ can be derived from level k in the Gaussian pyramid. That is, as stated in Chapter 11, level $k + 1$ in the Gaussian pyramid can be obtained through low-pass filtering applied to level k, followed by subsampling. Figure 13.8(c), level $k + 1$ is first interpolated, thus producing an estimate of level k, $\hat{f}^k(x, y)$. The difference between the original level k and the interpolated estimate of level k generates an error at level k, denoted by $e^k(x, y)$. If there are no quantization errors involved, then level k, $f^k(x, y)$ can be recovered completely from the interpolated estimate of level k, $\hat{f}^k(x, y)$, and the error at level k, $e^k(x, y)$. That is,

$$f^k(x,y) = \hat{f}^k(x,y) + e^k(x,y). \tag{13.20}$$

With quantization errors, however, the recovery of level k, $f^k(x, y)$ is not error free. It can be shown that coding $\hat{f}^k(x, y)$ and $e^k(x, y)$ is more efficient than directly coding $f^k(x, y)$.

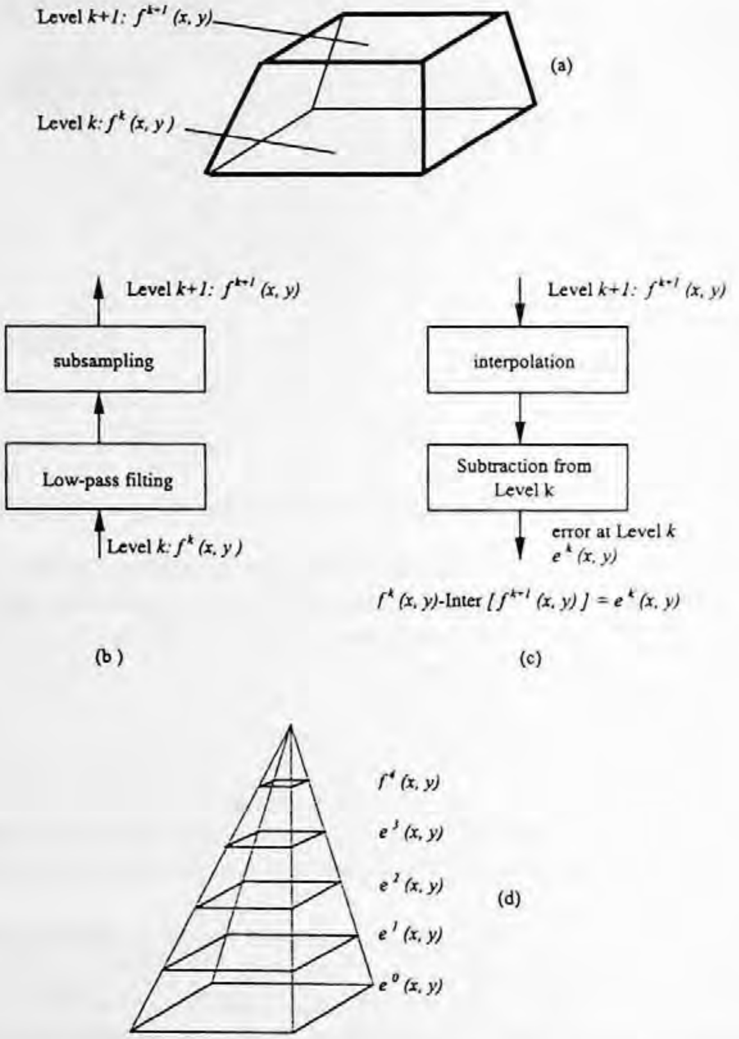


FIGURE 13.8 Laplacian pyramid (level k in a Gaussian pyramid). (a) Two consecutive levels in a pyramid structure. (b) Derivation of level $k + 1$ from level K. (c) Derivation of error at level k in a Laplacian pyramid. (d) Structure of Laplacian pyramid.

A set of images $e^k(x, y)$, $k = 0, 1, \ldots, K - 1$ and $f^K(x, y)$ forms a Laplacian pyramid. Figure 13.8(d) displays a Laplacian pyramid with $K = 5$. It can be shown that Laplacian pyramids provide an efficient way for image coding (Burt and Adelson, 1983). A more-detailed description of Gaussian and Laplacian pyramids can be found in Burt (1984) and Lim (1990).

13.3.2 The Singh Method

Singh (1991, 1992) presented a unified point of view on optical flow computation. He classified the information available in image sequences for optical flow determination into two categories: conservation information and neighborhood information. Conservation information is the information assumed to be conserved from one image frame to the next in flow estimation. Intensity is an example of conservation information, which is used most frequently in flow computation. Clearly, the brightness invariance constraint in the Horn and Schunck method is another way to state this type of conservation. Some functions of intensity may be used as conservation information as well.

In fact, Singh uses the Laplacian of intensity as conservation information for computational simplicity. More examples can be found later in Section 13.4. Other information, different from intensity, such as color, can be used as conservation information. Neighborhood information is the information available in the neighborhood of the pixel from which optical flow is estimated.

These two different types of information correspond to two steps in flow estimation. In the first step, conservation information is extracted, resulting in an initial estimate of flow vector. In the second step, this initial estimate is propagated into a neighborhood area and is iteratively updated. Obviously, in the Horn and Schunck method, the smoothness constraint is essentially one type of neighborhood information. Iteratively, estimates of flow vectors are refined with neighborhood information so that flow estimators from areas having sufficient intensity variation, such as the intensity corners as shown in Figure 13.2(d) and areas with strong texture, can be propagated into areas with relatively small intensity variation or uniform intensity distribution.

With this unified point of view on optical flow estimation, Singh treated flow computation as parameter estimation. By applying estimation theory to flow computation, he developed an estimation-theoretical method to determine optical flow. It is a correlation-based method and consists of the above-mentioned two steps.

13.3.2.1 Conservation Information

In the first step, for each pixel (x, y) in the current frame $f_n(x, y)$, a correlation window of $(2l + 1) \times (2l + 1)$ is opened, centered on the pixel. A search window of (2N+1) × (2N+1) is opened in the previous frame $f_{n-1}(x, y)$ centered on (x, y). An error distribution of those $(2N + 1) \times (2N + 1)$ samples are calculated by using SSD as follows:

$$E_c(u,v) = \sum_{s=-l}^{l} \sum_{t=-l}^{l} \left[f_n(x+s, y+t) - f_{n-1}(x-u+s, y-v+t) \right]^2 \quad -N \le u, v \le N. \tag{13.21}$$

A response–distribution for these $(2N + 1) \times (2N + 1)$ samples is then calculated.

$$R_c(u,v) = e^{-\beta E_c(u,v)}, \tag{13.22}$$

where β is a parameter, whose function and selection will be described in Section 13.3.3.1.

According to the weighted-least-square estimation, the optical flow can be estimated in this step as follows:

$$\begin{aligned} u_c &= \frac{\sum_u \sum_v R_c(u,v)u}{\sum_u \sum_v R_c(u,v)} \\ v_c &= \frac{\sum_u \sum_v R_c(u,v)v}{\sum_u \sum_v R_c(u,v)}. \end{aligned} \tag{13.23}$$

Assuming errors are additive and zero-mean random noise, we can also find the covariance matrix associated with the above estimate:

$$S_c = \begin{pmatrix} \dfrac{\sum_u \sum_v R_c(u,v)(u-u_c)^2}{\sum_u \sum_v R_c(u,v)} & \dfrac{\sum_u \sum_v R_c(u,v)(u-u_c)(v-v_c)}{\sum_u \sum_v R_c(u,v)} \\ \dfrac{\sum_u \sum_v R_c(u,v)(u-u_c)(v-v_c)}{\sum_u \sum_v R_c(u,v)} & \dfrac{\sum_u \sum_v R_c(u,v)(v-v_c)^2}{\sum_u \sum_v R_c(u,v)} \end{pmatrix}. \tag{13.24}$$

13.3.2.2 Neighborhood Information

After step 1, all initial estimates are available. In step 2, they need to be refined according to neighborhood information. For each pixel, the method considers a $(2w + 1) \times (2w + 1)$ neighborhood centered on it. The optical flow of the center pixel is updated from the estimates in the neighborhood. A set of Gaussian coefficients is used in the method such that the closer the neighbor pixel to the center pixel, the more influence the neighbor pixel has on the flow vector of the center pixel. The weighted-least-square based estimate in this step is

$$\begin{aligned} \bar{u} &= \frac{\sum_u \sum_v R_n(u,v)u}{\sum_u \sum_v R_n(u,v)} \\ \bar{v} &= \frac{\sum_u \sum_v R_n(u,v)v}{\sum_u \sum_v R_n(u,v)}, \end{aligned} \tag{13.25}$$

and the associated covariance matrix is

$$S_c = \begin{pmatrix} \dfrac{\sum_i R_n(u_i,v_i)(u_i-\bar{u})^2}{\sum_i R_n(u_i,v_i)} & \dfrac{\sum_i R_n(u_i,v_i)(u_i-\bar{u})(v_i-\bar{v})}{\sum_i R_n(u_i,v_i)} \\ \dfrac{\sum_i R_n(u_i,v_i)(u_i-\bar{u})(v_i-\bar{v})}{\sum_i R_n(u_i,v_i)} & \dfrac{\sum_i R_n(u_i,v_i)(v_i-\bar{v})^2}{\sum_i R_n(u_i,v_i)} \end{pmatrix}, \tag{13.26}$$

where $1 \leq i \leq (2w + 1)^2$.

In implementation, Singh uses a 3×3 neighborhood (i.e., $w = 1$) centered on the pixel under consideration. The weights are depicted in Figure 13.9.

13.3.2.3 Minimization and Iterative Algorithm

According to estimation theory (Beck and Arnold, 1977), two covariance matrices, expressed in Equations 13.24 and 13.26, respectively, are related to the confidence measure. That is, the reciprocals of the eigenvalues of the covariance matrix reveal confidence of the estimate along the

(0.25×0.25) $\frac{1}{16}$	(0.5×0.25) $\frac{1}{8}$	(0.25×0.25) $\frac{1}{16}$
(0.5×0.25) $\frac{1}{8}$	(0.5×0.5) $\frac{1}{4}$	(0.5×0.25) $\frac{1}{8}$
(0.25×0.25) $\frac{1}{16}$	(0.5×0.25) $\frac{1}{8}$	(0.25×0.25) $\frac{1}{16}$

FIGURE 13.9 3 × 3 Gaussian mask.

direction represented by the corresponding eigenvectors. Moreover, conservation error and neighborhood error can be represented as the following two quadratic terms, respectively.

$$\left(U-U_c\right)^T S_c^{-1}\left(U-U_c\right) \tag{13.27}$$

$$\left(U-\bar{U}\right)^T S_n^{-1}\left(U-\bar{U}\right), \tag{13.28}$$

where $\bar{U} = (\bar{u}, \bar{v})$, $U_c = (u_c, v_c)$, $U = (u, v)$.

The minimization of the sum of these two errors over the image area leads to an optimal estimate of optical flow. That is, find (u, v) such that the following error is minimized.

$$\sum_x \sum_y \left[\left(U-U_c\right)^T S_c^{-1}\left(U-U_c\right)+\left(U-\bar{U}\right)^T S_n^{-1}\left(U-\bar{U}\right)\right]. \tag{13.29}$$

An iterative procedure according to the Gauss–Siedel algorithm (Ralston and Rabinowitz, 1978) is used by Singh:

$$\begin{aligned} U^{k+1} &= \left[S_c^{-1}+S_n^{-1}\right]^{-1}\left[S_c^{-1}U_c+S_n^{-1}\bar{U}^k\right] \\ U^0 &= U_c. \end{aligned} \tag{13.30}$$

Note that U_c, S_c are calculated once and remain unchanged in all the iterations. On the contrary, $\bar{U}$ and S_n vary with each iteration. This agrees with the description of the method in Section 13.3.2.2.

13.3.3 The Pan, Shi, and Shu Method

Applying feedback (a powerful technique widely used in automatic control and many other fields) to a correlation-based algorithm, Pan, Shi, and Shu developed a correlation-feedback method to compute optical flow. The method is iterative in nature. In each iteration, the estimated optical flow and its several variations are fed back. For each of the varied optical flow vectors, the corresponding sum of squared displaced frame difference (DFD), which was discussed in Chapter 12 and which often involves bilinear interpolation, is calculated. This useful information is then utilized in a revised version of a correlation-based algorithm (Singh, 1992). They choose to work with this

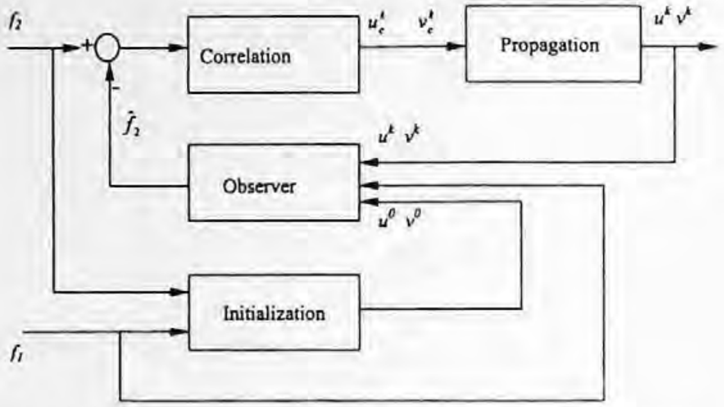


FIGURE 13.10 Block diagram of correlation feedback technique.

algorithm because it has several merits, and its estimation-theoretical computation framework lends itself to the application of the feedback technique.

As expected, the repeated usage of two given images via the feedback iterative procedure improves the accuracy of optical flow considerably. Several experiments on real image sequences in the laboratory and some synthetic image sequences demonstrate that the correlation-feedback algorithm performs better than some standard gradient- and correlation-based algorithms in terms of accuracy.

13.3.3.1 Proposed Framework

The block diagram of the proposed framework is shown in Figure 13.10 and described next.

Initialization — Although any flow algorithms can be used to generate an initial optical flow field $\bar{u}^o = (u^o, v^o)$ (even a nonzero initial flow field without applying any flow algorithm may work, but slowly), the Horn and Schunck algorithm (Horn and Schunck, 1981), discussed in Section 13.2.1 (usually 5 to 10 iterations) is used to provide an appropriate starting point after preprocessing (involving low-pass filtering), since the algorithm is fast and the problem caused by the smoothness constraint is not serious in the first 10 to 20 iterations. The modified Horn and Schunck method, discussed in Section 13.2.2, may also be used for the initialization.

Observer — The DFD at the kth iteration is observed as $f_n(\bar{x}) - f_{n-1}(\bar{x} - \bar{u}^k)$, where f_n and f_{n-1} denote two consecutive digital images, $\bar{x} = (x, y)$ denotes the spatial coordinates of the pixel under consideration, and $\bar{u}^k = (u^k, v^k)$ denotes the optical flow of this pixel estimated at the kth iteration. (Note that the vector representation of the spatial coordinates in image planes is used quite often in the literature, because of its brevity in notation.) Demanding fractional pixel accuracy usually requires interpolation. In the Pan et al. work, the bilinear interpolation is adopted. The bilinearly interpolated image is denoted by $\hat{f}_{n-1}$.

Correlation — Once the bilinearly interpolated image is available, a correlation measure needs to be selected to search for the best match of a given pixel in $f_n(\bar{x})$ in a search area in the interpolated image. In their work, the sum-of-square-differences (SSD) is used. For each pixel in f_n, a correlation window W_c of size $(2l + 1) \times (2l + 1)$ is formed, centered on the pixel.

The search window in the proposed approach is quite different from that used in the correlation-based approach, say, that of Singh (1992). Let u be a quantity chosen from the following five quantities:

$$u \in \left\{ u^k - \frac{1}{2}u^k, u^k - \frac{1}{4}u^n, u^k, u^k + \frac{1}{4}u^k, u^k + \frac{1}{2}u^k \right\}. \tag{13.31}$$

Let v be a quantity chosen from the following five quantities:

$$v \in \left\{ v^k - \frac{1}{2}v^k, v^k - \frac{1}{4}v^n, v^k, v^k + \frac{1}{4}v^k, v^k + \frac{1}{2}v^k \right\}. \tag{13.32}$$

Hence, there are 25 (i.e., 5 × 5) possible combinations for (u, v). (It is noted that the restriction of the nonzero initial flow field mentioned above in part A comes from here). Note that other choices of variations around (u^k, v^k) are possible. Each of them corresponds to a pixel, $(x - u, y - v)$, in the bilinearly interpolated image plane. A correlation window is formed and centered in this pixel. The 25 samples of error distribution around (u^k, v^k) can be computed by using the SSD. That is,

$$E(u,v) = \sum_{s=-l}^{l} \sum_{t=-l}^{l} \left(f_n(x+s, y+t) - \hat{f}_{n-1}(x-u+s, y-v+t) \right)^2. \tag{13.33}$$

The 25 samples of response distribution can be computed as follows:

$$R_c(u,v) = e^{-\beta E(u,v)}, \tag{13.34}$$

where β is chosen so as to make the maximum R_c among the 25 samples of response distribution be a number close to unity. The choice of an exponential function for converting the error distribution into the response distribution is based primarily on the following consideration: the exponential function is well behaved when the error approaches zero and all the response distribution values are positive. The choice of β mentioned above is motivated by the following observation: in this way, the R_c values, which are the weights used in Equation 13.35, will be more effective. That is, the computation in Equation 13.35 will be more sensitive to the variation of the error distribution defined in Equation 13.33.

The optical flow vector derived at this correlation stage is then calculated as follows, according to the weighted-least-squares estimation (Singh, 1992).

$$u^k(x,y) = \frac{\sum_u \sum_v R_c(u,v)u}{\sum_u \sum_v R_c(u,v)}, \quad v_c^k(x,y) = \frac{\sum_u \sum_v R_c(u,v)v}{\sum_u \sum_v R_c(u,v)}. \tag{13.35}$$

Propagation — Except in the vicinity of motion boundaries, the motion vectors associated with neighboring pixels are expected to be similar. Therefore, this constraint can be used to regularize the motion field. That is,

$$u^{k+1}(x,y) = \sum_{i=-w}^{w} \sum_{j=-w}^{w} w_1(i,j) u_c^k(x+i, y+j), v^{k+1}(x,y) = \sum_{i=-w}^{w} \sum_{j=-w}^{w} w_1(i,j) u_c^k(x+i, y+j), \tag{13.36}$$

where $w_1(i, j)$ is a weighting function. The Gaussian mask shown in Figure 13.9 is chosen as the weighting function $w_1(i, j)$ used in our experiments. By using this mask, the velocity of various pixels in the neighborhood of a pixel will be weighted according to their distance from the pixel: the larger the distance, the smaller the weight. The mask smooths the optical flow field as well.

Convergence — Under the assumption of the symmetric response distribution with a single maximum value assumed by the ground-truth optical flow, the convergence of the correlation-feedback technique is justified by Pan et al. (1995).

13.3.3.2 Implementation and Experiments

Implementation — To make the algorithm more robust against noise, three consecutive images in an image sequence, denoted by f_1, f_2, and f_3, respectively, are used to implement their algorithm instead of the two images in the above principle discussion. This implementation was proposed by Singh (1992). Assume the time interval between f_1 and f_2 is the same as that between f_2 and f_3. Also assume the apparent 2-D motion is uniform during these two intervals along the motion trajectories. From images f_1 and f_2, (u^o, v^o) can be computed. From (u^k, v^k), the optical flow estimated during the kth iteration, and f_1 and f_2, the response distribution, $R_c^+(u^k, v^k)$, can be calculated as

$$R_c^+\left(u^k,v^k\right)=\exp\left\{-\beta\sum_{s=-l}^{l}\sum_{t=-l}^{l}\left[f_2(x+s,y+t)-\hat{f}_1\left(x-u^k+s,y-v^k+t\right)\right]^2\right\}. \tag{13.37}$$

Similarly, from images f_3 and f_2, $(-u^k, -v^k)$ can be calculated. Then $R_c^-(-u^k, -v^k)$ can be calculated as

$$R_c^-\left(-u^k,-v^k\right)=\exp\left\{-\beta\sum_{s=-l}^{l}\sum_{t=-l}^{l}\left[f_2(x+s,y+t)-\hat{f}_3\left(x-u^k+s,y+v^k+t\right)\right]^2\right\}. \tag{13.38}$$

The response distribution $R_c(u^k, v^k)$ can then be determined as the sum of $R_c^+(u^k, v^k)$ and $R_c^-(-u^k,-v^k)$. The size of the correlation window and the weighting function is chosen to be 3 × 3, i.e., $l = 1$, $w = 1$. In each search window, β is chosen so as to make the larger one among R_c^+ and R_c^- a number close to unity. In the observer stage, the bilinear interpolation is used, which is shown to be faster and better than the B-spline in the many experiments of Pan et al.

Experiment I — Figure 13.11 shows the three successive image frames f_1, f_2, and f_3 about a square post. They were taken by a CCD video camera and a DATACUBE real-time image processing system supported by a Sun workstation. The square post is moving horizontally, perpendicular to the optical axis of the camera, in a uniform speed of 2.747 pixels per frame. To remove various noises to a certain extent and to speed up processing, these three 256 × 256 images are low-pass filtered and then subsampled prior to optical flow estimation. That is, the intensities of every 16 pixels in a block of 4 × 4 are averaged and the average value is assigned to represent this block. Note that the choice of other low-pass filters is also possible. In this way, these three images are compressed into three 64 × 64 images. The "ground-truth" 2-D motion velocity vector is hence known as $u^a = -0.6868$; $v^a = 0$.

To compare the performance of the correlation-feedback approach with that of the gradient-based and correlation-based approaches, the Horn and Schunck algorithm is chosen to represent the gradient-based approach and Singh's framework to represent the correlation-based approach. Table 13.1 shows the results of the comparison. There, l, w, and N indicate the sizes of the correlation window, weighting function, and search window, respectively. The program that implements Singh's algorithm is provided by Barron et al. (1994). In the correlation-feedback algorithm, ten iterations of the Horn and Schunck algorithm with $\alpha = 5$ are used in the initialization. (Recall that the α is a regularization parameter used by Horn and Schunck, 1981). Only the central 40 × 40 flow vector array is used to compute u_{error}, which is the root mean square (RMS) error in the vector magnitudes between the ground-truth and estimated optical flow vectors. It is noted that the relative error in Experiment I is greater than 10%. This is because the denominator in the formula calculating the RMS error is too small due to the static background and, hence, there are many zero ground-truth 2-D motion velocity vectors in this experiment. Relatively speaking, the correlation-feedback algorithm performs best in determining optical flow for a texture post in translation. The correct optical flow field and those calculated by using three different algorithms are shown in Figure 13.12.

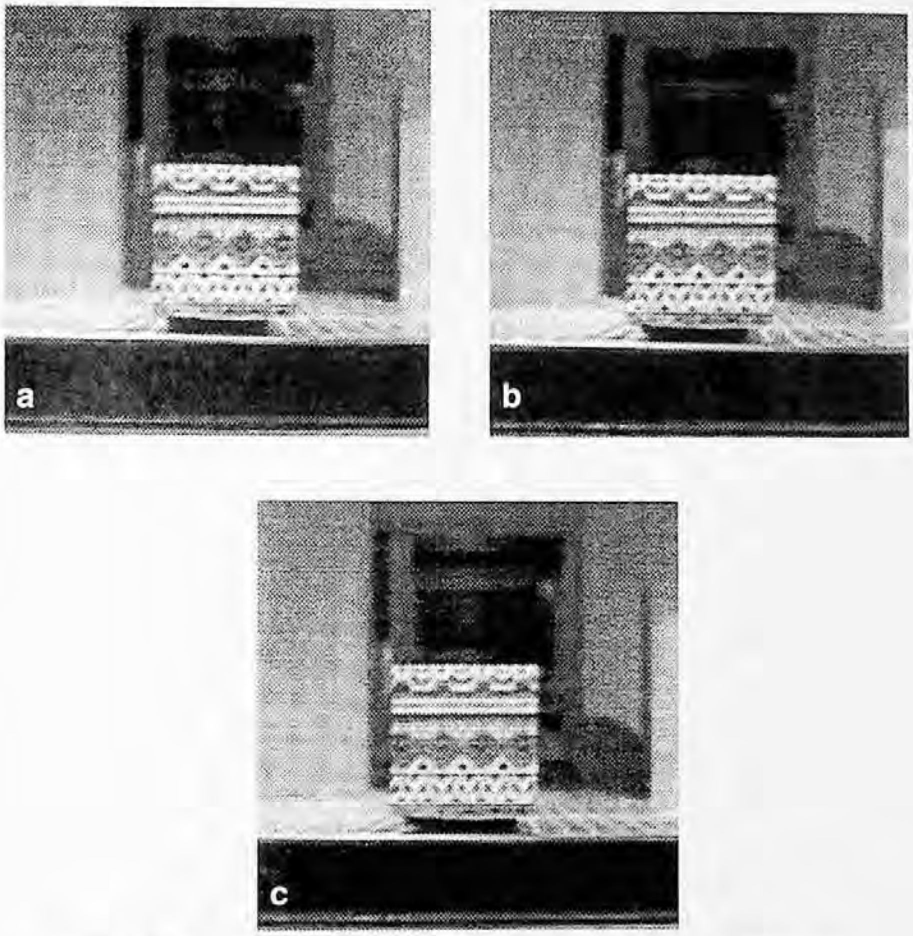
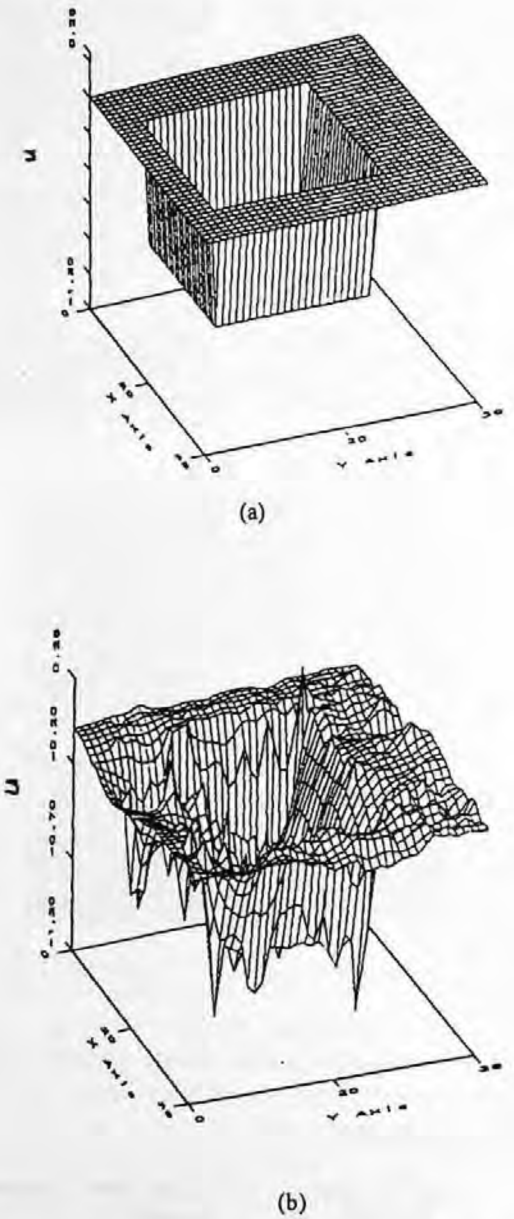


FIGURE 13.11 Texture square (a). Texture square (b). Texture square (c).

TABLE 13.1
Comparison in Experiment I

Techniques	Gradient-Based Approach	Correlation-Based Approach	Correlation-Feedback Approach
13.3.3.3 Conditions	*Iteration no.* = 128 $\alpha = 5$	*Iteration no.* = 25 $l = 2$, $w = 2$ N = 4	*Iteration no.* = 10 *Iteration no.* (*Horn*) = 10 $l = 1$, $w = 1$, $N = 5$
u_{error}	56.37%	80.97%	44.56%

Experiment II — The images in Figure 13.13 were obtained by rotating a CCD camera with respect to the center of a ball. The rotating velocity is 2.5° per frame. Similarly, three 256 × 256 images are compressed into three 64 × 64 images by using the averaging and subsampling discussed above. Only the central 40 × 40 optical vector arrays are used to compute u_{error}. Table 13.2 reports the results for this experiment. There, u_{error}, l, w, and N have the same meaning as that discussed in Experiment I. It is obvious that our correlation-feedback algorithm performs best in determining optical flow for this rotating ball case.



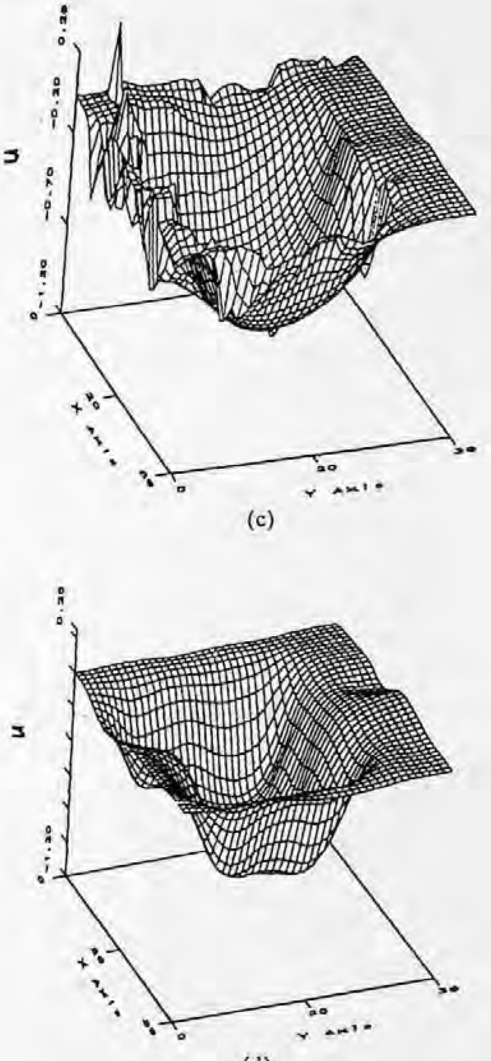
(a)

(b)

FIGURE 13.12 (a) Correct optical flow field. (b) Optical flow field calculated by the gradient-based approach. (c) Optical flow field calculated by the correlation-based approach. (d) Optical flow field calculated by the correlation-feedback approach.

Experiment III — To compare the correlation-feedback algorithm with other existing techniques in a more objective, quantitative manner, Pan et al. cite some results reported by Barron et al. (1994), which were obtained by applying some typical optical flow techniques to some image sequences chosen with deliberation. In the meantime they report the results obtained by applying their feedback technique to the identical image sequences with the same accuracy measurement as used by Barron et al. (1994).

Three image sequences used by Barron et al. (1994) were utilized here. They are named "Translating Tree," "Diverging Tree," and "Yosemite." The first two simulate translational camera motion with respect to a textured planar surface (Figure 13.14), and are sometimes referred to as



(c)

(d)

FIGURE 13.12 (continued)

"Tree 2-D" sequence. Therefore, there are no occlusions and no motion discontinuities in these two sequences. In the "Translating Tree" sequence, the camera moves normally to its line of sight, with velocities between 1.73 and 2.26 pixels/frame parallel to the x-axis in the image plane. In the "Diverging Tree" sequence, the camera moves along its line of sight. The focus of expansion is at the center of the image. The speeds vary from 1.29 pixels/frame on left side to 1.86 pixels/frame on the right. The "Yosemite" sequence is a more complex test case (see Figure 13.15). The motion in the upper right is mainly divergent. The clouds translate to the right with a speed of 1 pixel/frame, while velocities in the lower left are about 4 pixels/frame. This sequence is challenging because of the range of velocities and the occluding edges between the mountains and at the horizon. There is severe aliasing in the lower portion of the images, causing most methods to produce poorer velocity measurements. Note that this synthetic sequence is for quantitative study purposes since its ground-truth flow field is known and is, otherwise, far less complex than many real-world outdoor sequences processed in the literature.

The angular measure of the error used by Barron et al. (1994) is utilized here, as well. Let image velocity $\bar{u} = (u, v)$ be represented as 3-D direction vectors,

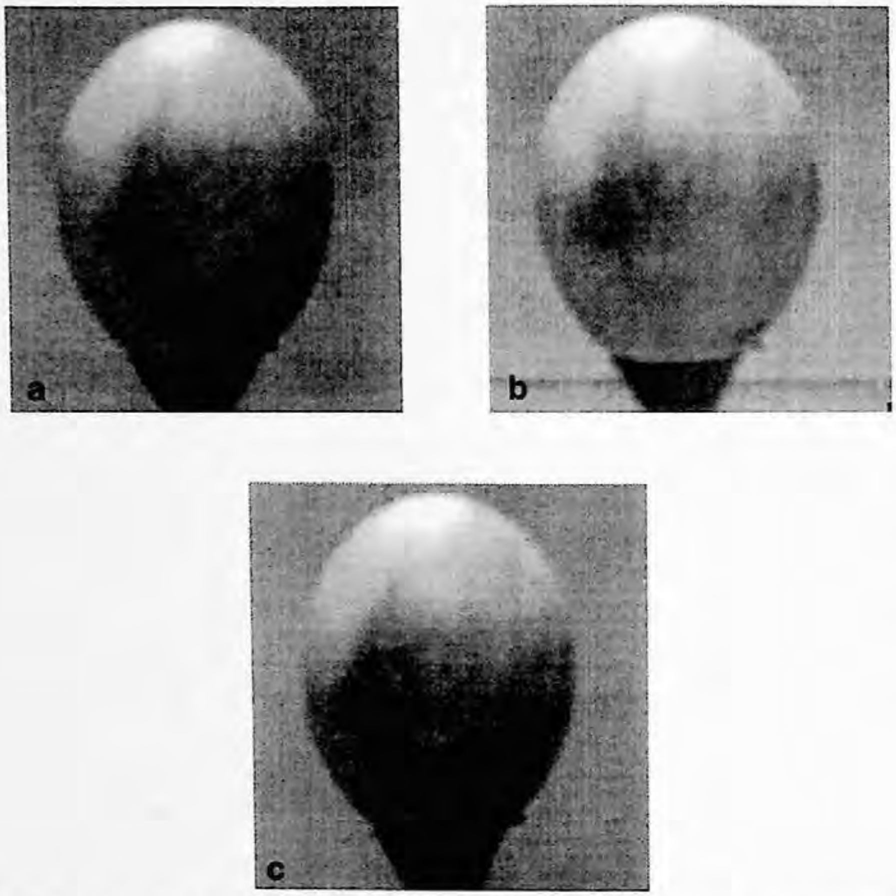


FIGURE 13.13 A rotating ball in three different frames — a, b, c. The rotating velocity is 2.5° per frame.

TABLE 13.2
Comparison in Experiment II

Techniques	Gradient-Based Approach	Correlation-Based Approach	Correlation-Feedback Approach
Conditions	*Iteration no.* = 128 $\alpha = 5$	*Iteration no.* = 25 $l = 2, w = 2$ N = 4	*Iteration no.* = 10 *Iteration no.* (*Horn*) = 10 $l = 1, w = 1, N = 5$
u_{error}	65.67%	55.29%	49.80%

$$\bar{V} \equiv \frac{1}{\sqrt{u^2 + v^2 + 1}}(u, v, 1). \tag{13.39}$$

The angular error between the correct image velocity $\bar{V}$ and an estimate $\bar{V}_e$ is ψ_E = across $(\bar{V}_c \cdot \bar{V}_e)$. It is obvious that the smaller the angular error ψ_E, the more accurate the estimation of the optical flow field will be. Despite the fact that the confidence measurement can be used in the correlation-feedback algorithm, as well, Pan et al. did not consider the usage of the confidence measurement in their work. Therefore, only the results with 100% density in Tables 4.6, 4.7, and 4.10 in the Barron et al. (1994) paper were used in Tables 13.3, 13.4, and 13.5, respectively.

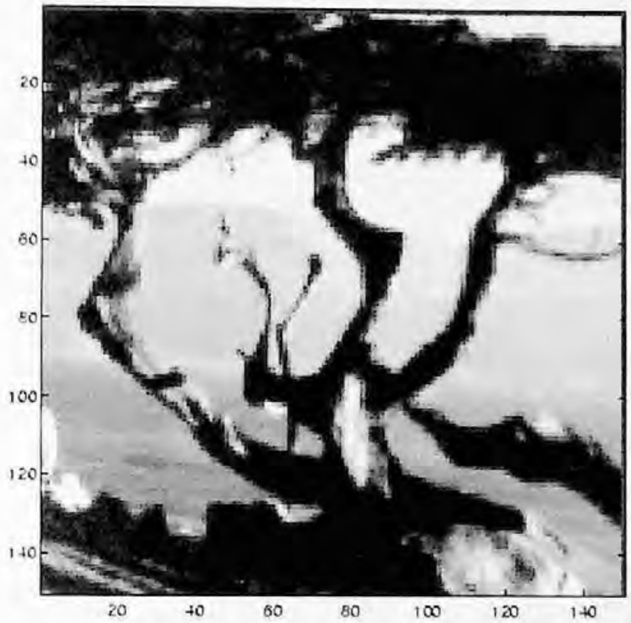


FIGURE 13.14 A frame of the "Tree 2-D" sequence.

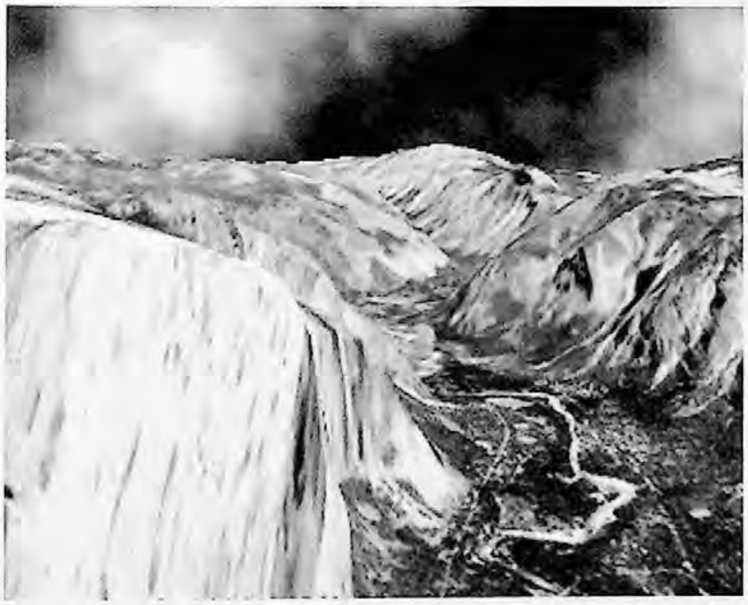


FIGURE 13.15 A frame of the "Yosemite" sequence.

Prior to computation of the optical flow field, the "Yosemite" and "Tree 2-D" test sequences were compressed by a factor of 16 and 4, respectively, using the averaging and subsampling method discussed earlier.

As mentioned by Barron et al. (1994) the optical flow field for the "Yosemite" sequence is complex, and Table 13.5 indicates that the correlation-feedback algorithm evidently performs best. A robust method was developed and applied to a cloudless Yosemite sequence (Black and Anandan, 1996). It is noted that the performance of flow determination algorithms will be improved if the sky is removed from consideration (Barron et al., 1994; Black and Anandan, 1996). Still, it is clear

TABLE 13.3
Summary of the "Translating Tree" 2-D Velocity Results

Techniques	Average Error, °	Standard Deviation, °	Density, %
Horn and Schunck (original)	38.72	27.67	100
Horn and Schunck (modified)	2.02	2.27	100
Uras et al. (unthresholded)	0.62	0.52	100
Nagel	2.44	3.06	100
Anandan	4.54	3.10	100
Singh (step 1, $l = 2$, $w = 2$)	1.64	2.44	100
Singh (step 2, $l = 2$, $w = 2$)	1.25	3.29	100
Correlation feedback ($l = 1$, $w = 1$)	1.07	0.48	100

TABLE 13.4
Summary of the "Diverging Tree" 2-D Velocity Results

Techniques	Average Error, °	Standard Deviation, °	Density, %
Horn and Schunck (original)	12.02	11.72	100
Horn and Schunck (modified)	2.55	3.67	100
Uras et al. (unthresholded)	4.64	3.48	100
Nagel	2.94	3.23	100
Anandan (frames 19 and 21)	7.64	4.96	100
Singh (step 1, $l = 2$, $w = 2$)	17.66	14.25	100
Singh (step 2, $l = 2$, $w = 2$)	8.60	5.60	100
Pan, Shi, and Shu ($l = 1$, $w = 1$)	5.12	2.16	100

TABLE 13.5
Summary of the "Yosemite" 2-D Velocity Results

Techniques	Average Error, °	Standard Deviation, °	Density, %
Horn and Schunck (original)	32.43	30.28	100
Horn and Schunck (modified)	11.26	16.41	100
Uras et al. (unthresholded)	10.44	15.00	100
Nagel	11.71	10.59	100
Anandan (frames 19 and 21)	15.84	13.46	100
Singh (step 1, $l = 2$, $w = 2$)	18.24	17.02	100
Singh (step 2, $l = 2$, $w = 2$)	13.16	12.07	100
Pan, Shi, and Shu ($l = 1$, $w = 1$)	7.93	6.72	100

that the algorithm in the Black and Anandan (1996) paper achieved very good performance in terms of accuracy. In order to make a comparison with their algorithm, the correlation-feedback algorithm was applied to the same cloudless Yosemite sequence. The results were reported in Table 13.6, from which it can be observed that the results obtained by Pan et al. are slightly better. Tables 13.3 and 13.4 indicate that the feedback technique also performs very well in translating and diverging texture post cases.

Experiment IV — Here, the correlation-feedback algorithm is applied to a real sequence named *Hamburg Taxi*, which is used as a testing sequence by Barron et al. (1994). There are four moving

TABLE 13.6
Summary of the cloudless "Yosemite" 2-D Velocity Results

Techniques	Average Error, °	Standard Deviation, °	Density, %
Robust formulation	4.46	4.21	100
Pan, Shi, and Shu (l = 1, w = 1)	3.79	3.44	100

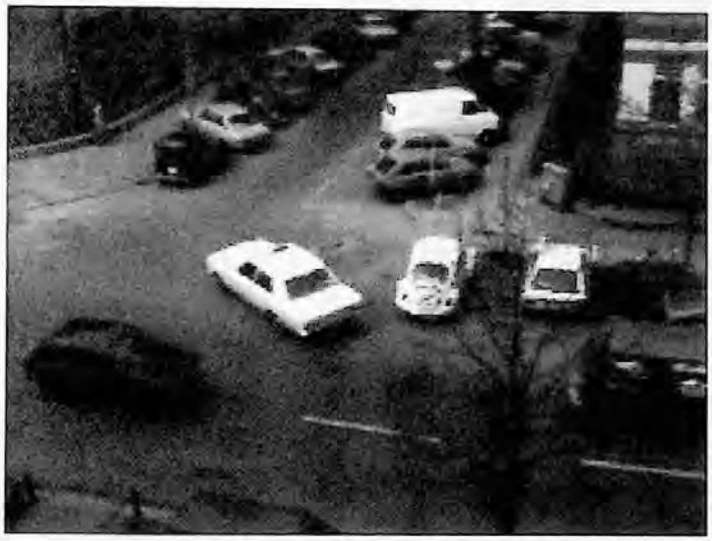


FIGURE 13.16 Hamburg Taxi.

objects in the scene: a moving pedestrian in the upper left portion, a turning car in the middle, a car moving toward right at the left side and a car moving toward left at the right side. A frame of the sequence and the needle diagram of flow vectors estimated by using ten iterations of the correlation-feedback algorithm (with ten iterations of the Horn and Schunck algorithm for initialization) are shown in Figures 13.16 and 13.17, respectively. The needle diagram is printed in the same fashion as those shown by Barron et al. (1994). It is noted that the moving pedestrian in the upper left portion cannot be shown because of the scale used in the needle diagram. The other three moving vehicles in the sequence are shown very clearly. The noise level is low. Compared with those diagrams reported by Barron et al. (1994), the correlation-feedback algorithm achieves very good results.

For a comparison on a local basis, the portion of the needle diagram associated with the area surrounding the turning car (a sample of the velocity fields), obtained by 50 iterations of the correlation-feedback algorithm with five iterations of the Horn and Schunck algorithm as initialization, is provided in Figure 13.18(c). Its counterparts obtained by applying the Horn and Schunck (50 iterations) and the Singh (50 iterations) algorithms are displayed in Figure 13.18(a) and (b), respectively. It is observed that the correlation-feedback algorithm achieves the best results among the three algorithms.

13.3.3.4 Discussion and Conclusion

Although it uses a revised version of a correlation-based algorithm (Singh, 1992), the correlation-feedback technique is quite different from the correlation-based algorithm (Singh, 1992) in the following four aspects. First, different optimization criteria: the algorithm does not use the iterative minimization procedure used in (Singh, 1992). Instead, some variations of the estimated optical

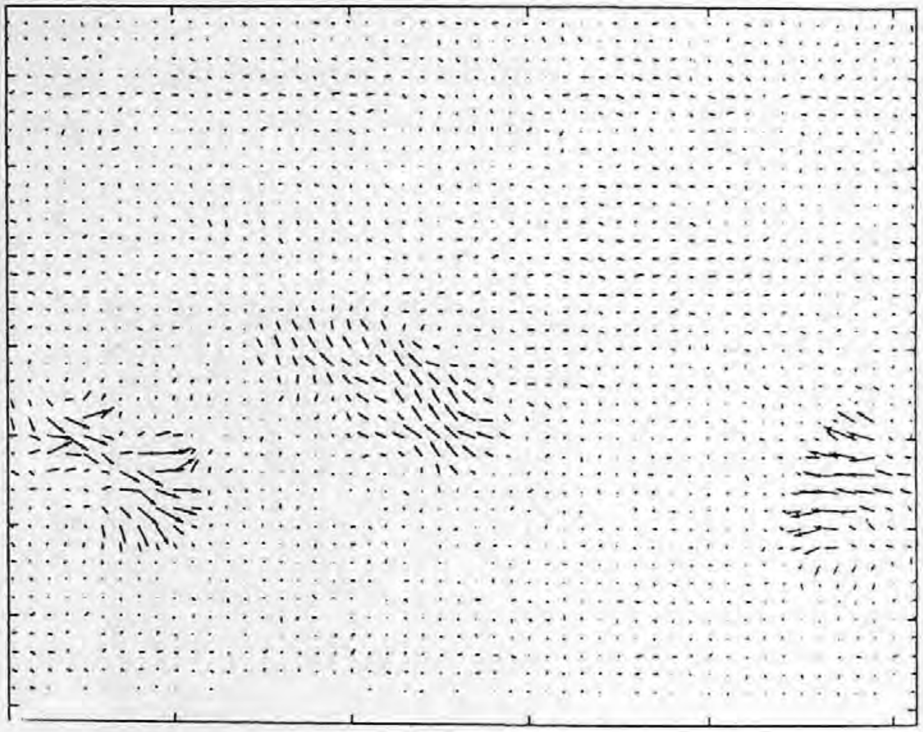
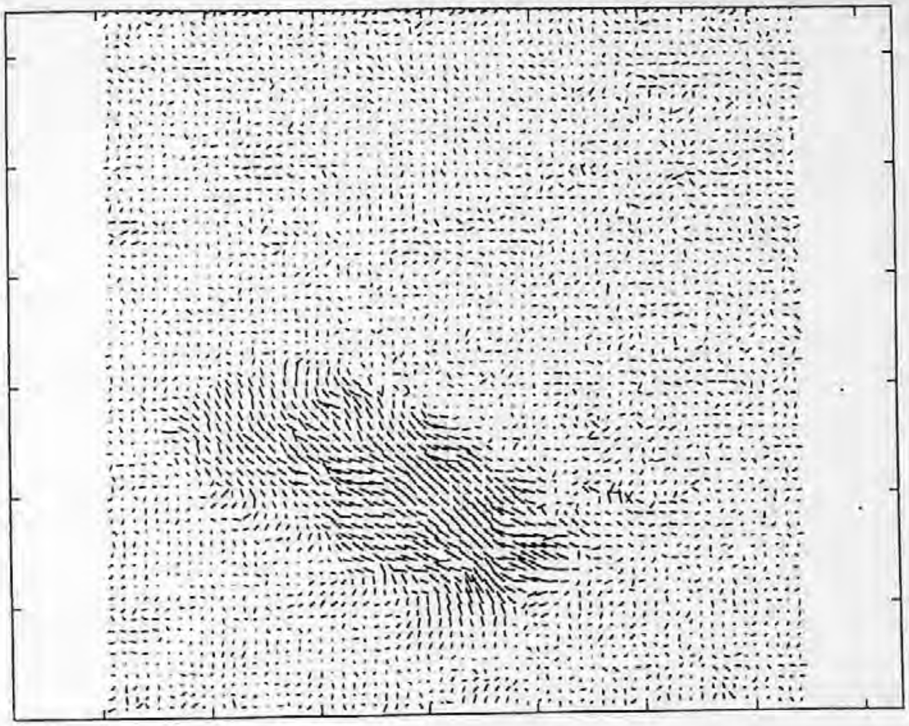


FIGURE 13.17 Needle diagram of flow field of Hamburg Taxi sequence obtained by using the correlation-feedback algorithm.

flow vectors are generated and fed back. The associated bilinearly interpolated displaced frame difference for each variation is calculated and utilized. In essence, the feedback approach utilizes two given images repeatedly, while the Singh method uses two given images only once (u_c and v_c derived from the two given images are only calculated once). The best local matching between the displaced image, generated via feedback of the estimated optical flow, and the given image is actually used as the ultimate criterion for improving optical flow accuracy in the iterative process. Second, the search window in the algorithm is an adaptive "rubber" window, having a variable size depending on (u^k, v^k). In the correlation-based approaches (Singh, 1992), the search window has a fixed size. Third, the algorithm uses a bilinear interpolation technique in the observation stage and provides the correlation stage with a virtually continuous image field for more accurate motion vector computation, while that of Singh (1992) does not. Fourth, different performances are achieved when image intensity is a linear function of image coordinates. In fact, in the vicinity of a pixel, the intensity can usually be considered as such a linear function. Except if the optical flow vectors happen to have only an integer multiple of pixels as their components, an analysis by Pan (1994) shows that the correlation-based approach (Singh, 1992) will not converge to the apparent 2-D motion vectors and will easily have error much greater than 10%. Pan (1994) also shows that the linear intensity function guarantees the assumption of the symmetric response distribution with a single maximum value assumed by the ground-truth optical flow. As discussed in Section 13.3.3.1, under this assumption the convergence of the correlation-feedback technique is justified.

Numerous experiments have demonstrated the convergence and accuracy of the correlation-feedback algorithm, and usually it is more accurate than some standard gradient- and correlation-based approaches. In the complicated optical flow cases, specifically in the case of the "Yosemite" image sequence (regarded as the most challenging quantitative test image sequence by Barron et al. (1994), it performs better than all other techniques.



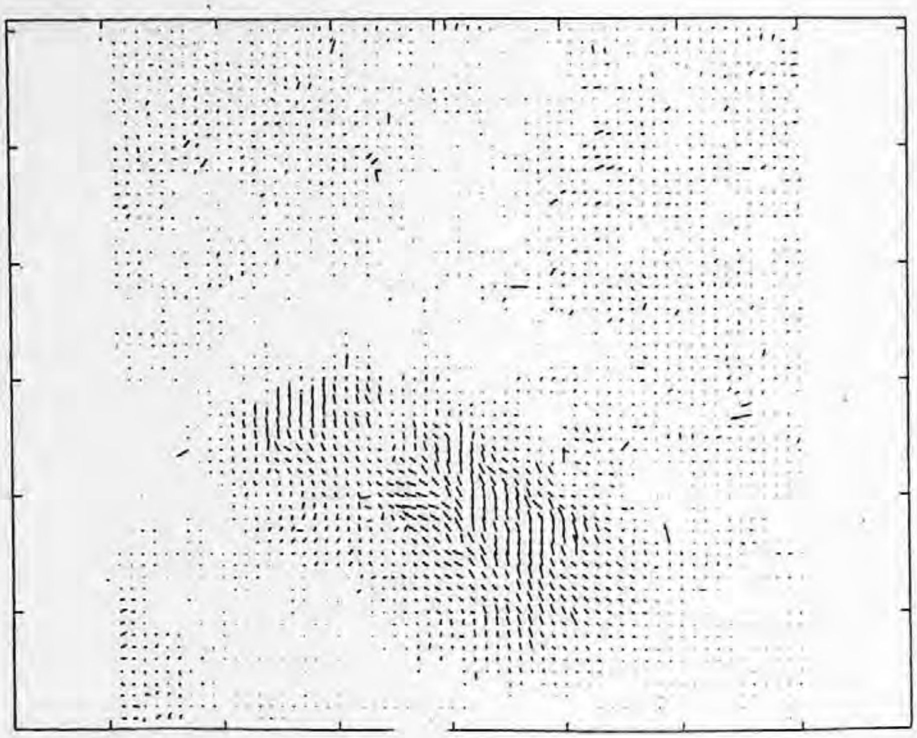
(a)

FIGURE 13.18 A portion of the needle diagram obtained by using (a) the Horn and Schunk algorithm, (b) the Singh algorithm, and (c) the correlation-feedback algorithm.

13.4 MULTIPLE ATTRIBUTES FOR CONSERVATION INFORMATION

As stated at the beginning of this chapter, there are many algorithms in optical flow computation reported in the literature. Many more new algorithms continue to be developed. In Sections 13.2 and 13.3, we introduced some typical algorithms using gradient- and correlation-based approaches. We will not explore various algorithms any further here. It is hoped that the fundamental concepts and algorithms introduced above have provided a solid base for readers to study more-advanced techniques.

We would like to discuss optical flow from another point of view, however: multiple image attributes vs. a single image attribute. All of the methods we have discussed so far use only one kind of image attributes as conservation information in flow determination. Most methods use intensity. Singh's method uses the Laplacian of intensity, which is calculated by using the difference of the Gaussian operation (Burt, 1984). It was reported by Weng, Ahuja, and Huang (1992) that using a single attribute as conservation information may result in ambiguity in matching two perspective views, while multiple attributes, which are motion insensitive, may reduce ambiguity remarkably, resulting in better matching. An example is shown in Figure 13.19 to illustrate this argument. In this section, the Weng et al. method is discussed first. Then we introduce the Xia and Shi method, which uses multiple attributes in a framework based on weighted-least-square estimation and feedback techniques.



(b)

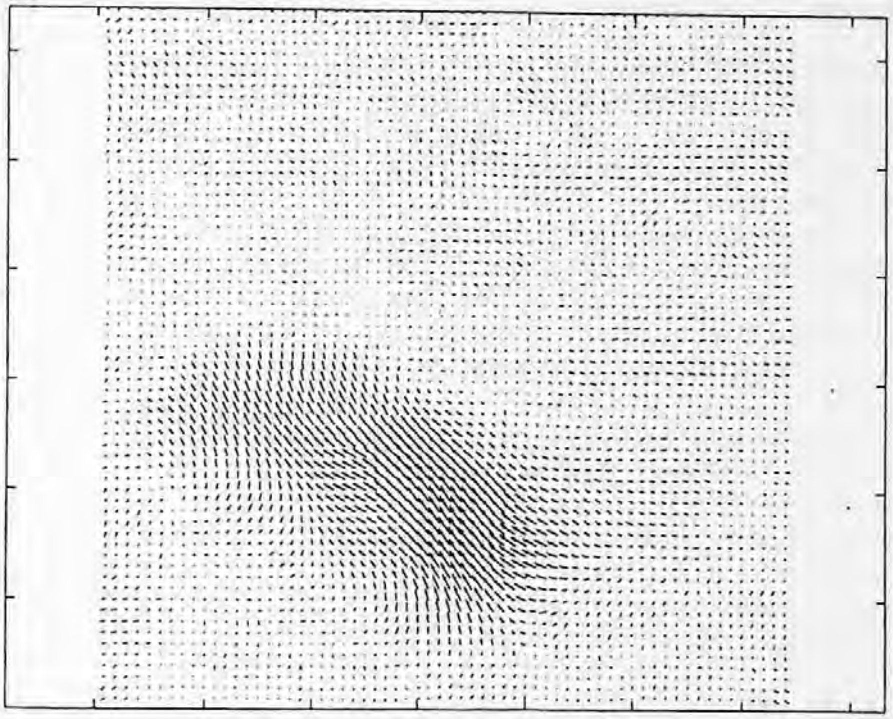
FIGURE 13.18 (continued)

13.4.1 The Weng, Ahuja, and Huang Method

Weng, Ahuja, and Huang proposed a quite different approach to image point matching (Weng et al., 1992). Note that the image matching amounts to flow field computation since it calculates a displacement field for each point in image planes, which is essentially a flow field if the time interval between two image frames is known.

Based on an analysis indicating that using image intensity as a single attribute is not enough in accurate image matching, Weng, Ahuja, and Huang utilize multiple attributes associated with images in estimation of the dense displacement field. These image attributes are motion insensitive; i.e., they generally sustain only small change under motion assumed to be locally rigid. The image attributes used are image intensity, edgeness, and cornerness. For each image attribute, the algorithm forms a residual function, reflecting the inaccuracy of the estimated matching. The matching is then determined via an iterative procedure to minimize the weighted sum of these residual functions. In handling neighborhood information, a more-advanced smoothness constraint is used to take care of moving discontinuities. The method considers uniform regions and the occlusion issue as well.

In addition to using multiple image attributes, the method is pointwise processing. There is no need for calculation of correlation within two correlation windows, which saves computation dramatically. However, the method also has some drawbacks. First, the edgeness and cornerness involve calculation of the spatial gradient, which is noise sensitive. Second, in solving for minimization, the method resorts to numerical differentiation again: the estimated displacement vectors are updated based on the partial derivatives of the noisy attribute images. In a word, the computational framework heavily relies on numerical differentiation, which is considered to be impractical for accurate computation (Barron et al., 1994).



(c)

FIGURE 13.18 (continued)

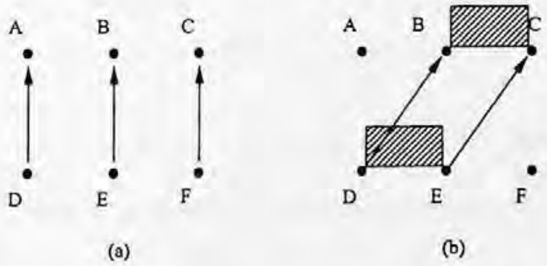


FIGURE 13.19 Multiple attributes vs. single attribute. (a) With intensity information only, points D, E, and F tend to match to points A, B, and C, respectively. (b) With intensity, edge and corner information points D and E tend to match points B and C, respectively.

On the other hand, the Pan, Shi, and Shu method, discussed in Section 13.3.3 in the category of correlation based approaches, seems to have some complementary features. It is correlation-based. It uses intensity as a single attribute. In these two aspects the Pan et al. method is inferior to the method by Weng, Ahuja, and Huang. The feedback technique and the weighted least-square computation framework used in the Pan et al. method are superior, however, compared with the method by Weng et al. Motivated by the above observations, an efficient, multiattribute feedback method was developed by Xia and Shi (Xia and Shi, 1995; Xia, 1996), and is discussed in the next subsection. It is expected that more insight into the Weng, Ahuja, and Huang method will become clear in the discussion as well.

13.4.2 The Xia and Shi Method

This method uses multiple attributes that are motion insensitive. The following five attributes are used: image intensity, horizontal edgeness, vertical edgeness, contrast, and entropy. The first three are used by Weng et al. (1992) as well, and can be considered as structural attributes, while the last two, which are not used by Weng et al. (1992), can be considered as textural attributes according to Haralick (1979).

Instead of the computational framework presented by Weng et al. (1992), which, as discussed above, may not be practical for accurate computation, the method uses the computational framework of Pan (1994; 1998). That is, the weighted-least-squared estimation technique used by Singh (1992) and the feedback technique used by Pan (1994; 1998) are utilized here. Unlike in the Weng et al. (1992) method, subpixel accuracy is considered and a confidence measure is generated in the method.

The Xia and Shi method is also different from those algorithms presented by Singh (1992) and Pan et al. (1995; 1998). First, there is no correlation in the method, while both Singh (1992) and Pan et al. (1995; 1998) are correlation based. Specifically, the method is a point-wise processing. Second, the method uses multiple attributes, while both Singh (1992) and Pan et al. (1995; 1998) use image intensity as a single attribute.

In summary, the Xia and Shi method to compute optical flow is motivated by several existing algorithms mentioned above. It does, however, differ from each of them significantly.

13.4.2.1 Multiple Image Attributes

As mentioned before, there are five image attributes in the Xia and Shi method. They are defined below.

Image Intensity — The intensity at a pixel (x, y) in an image $f_n(x, y)$, denoted by $A_i(x, y)$, i.e., $A_i(x, y) = f_n(x, y)$.

Horizontal Edgeness — The horizontal edgeness at a pixel (x, y), denoted by $A_h(x, y)$, is defined as

$$A_h(x,y)=\frac{\partial f(x,y)}{\partial y}, \tag{13.40}$$

i.e., the partial derivative of $f(x, y)$ with respect to y, the second component of the gradient of intensity function at the pixel.

Vertical Edgeness — The vertical edgeness at a pixel (x, y), denoted by $A_v(x, y)$, is defined as

$$A_v(x,y)=\frac{\partial f(x,y)}{\partial x}, \tag{13.41}$$

i.e., the first component of the gradient of intensity function at the pixel. Note that the partial derivatives in Equations 13.40 and 13.41 are computed by applying a Sobel operator (Gonzalez and Woods, 1992) in a 3 × 3 neighborhood of the pixel.

Contrast — The local contrast at a pixel (x, y), denoted by $A_c(x, y)$, is defined as

$$A_c(x,y)=\sum_{i,j\in S}(i-j)^2 C_{i,j}, \tag{13.42}$$

where S is a set of all the distinct gray levels within a 3×3 window centered at pixel (x, y). $C_{i,j}$ specifies a relative frequency with which two neighboring pixels separated horizontally by a distance of 1 occur in the 3×3 window, one with gray level i and the other with gray level j.

Entropy — The local entropy at a point (x, y), denoted by $A_e\ (x, y)$, is given by

$$A_e(x,y) = -\sum_{i \in S} p_i \log p_i, \tag{13.43}$$

where S was defined above, and p_i is the probability of occurrence of the gray level i in the 3×3 window.

Since the intensity is assumed to be invariant to motion, so are the horizontal edgeness, vertical edgeness, contrast, and entropy.

As mentioned above, the intensity and edgeness are used as attributes in the Weng et al. algorithm as well. Compared with the negative and positive cornerness used in the Weng et al. algorithm, the local contrast and entropy need no differentiation and therefore are less sensitive to various noises in original images. In addition, these two attributes are inexpensive in terms of computation. They reflect the textural information about the local neighborhood of the pixel for which the flow vector is to be estimated.

13.4.2.2 Conservation Stage

In the Xia and Shi algorithm, this stage is similar to that in the Pan et al. algorithm. That is, for a flow vector estimated at the kth iteration, denoted by (u^k, v^k), we find its 25 variations, (u, v), according to

$$\begin{aligned} u &\in \left\{ u^k - \frac{u^k}{2}, u^k - \frac{u^k}{4}, u^k, u^k + \frac{u^k}{4}, u^k + \frac{u^k}{2} \right\} \\ v &\in \left\{ v^k - \frac{v^k}{2}, v^k - \frac{v^k}{4}, u^k, v^k + \frac{v^k}{4}, v^k + \frac{v^k}{2} \right\}. \end{aligned} \tag{13.44}$$

For each of these 25 variations, the matching error is computed as

$$E(u,v) = r_{A_i}^2(x,y,u,v) + r_{A_h}^2(x,y,u,v) + r_{A_v}^2(x,y,u,v) + r_{A_c}^2(x,y,u,v) + r_{A_e}^2(x,y,u,v), \tag{13.45}$$

where $r_{A_i}, r_{A_h}, r_{A_v}, r_{A_c}, r_{A_e}$ denote the residual function with respect to the five attributes, respectively.

The residual function of intensity is defined as

$$r_{A_i}(x,y,u,v) = A_{i_n}(x,y) - A_{i_{n-1}}(x-u, y-v) = f_n(x,y) - f_{n-1}(x-u, y-v), \tag{13.46}$$

where $f_n\ (x, y)$, $f_{n-1}\ (x, y)$ is defined as before, i.e., the intensity function at t_n and t_{n-1}, respectively; A_{i_n}, $A_{i_{n-1}}$ denote the intensity attributes on f_n and f_{n-1}, respectively.

It is observed that the residual error of intensity is essentially the DFD discussed in Chapter 12. The rest of the residual functions are defined similarly. When subpixel accuracy is required, spatial interpolation in the attribute images generally is necessary. Thus, the flow vector estimation is now converted to a minimization problem. That is, find u and v at pixel (x, y) such that the matching error defined in Equation 13.45 is minimized. The weighted least-square method (Singh, 1992; Pan et al., 1998) is then used. That is,

$$R(u,v) = e^{-\beta E(u,v)} \tag{13.47}$$

$$u_c^{k+1} = \frac{\sum_u \sum_v R(u,v)u}{\sum_u \sum_v R(u,v)}, \quad v_c^{k+1} = \frac{\sum_u \sum_v R(u,v)v}{\sum_u \sum_v R(u,v)}. \tag{13.48}$$

Since the weighted least-square method has been discussed in detail in Sections 13.3.2 and 13.3.3, we will not go into more detail here.

13.4.2.3 Propagation Stage

Similar to what was proposed in the Pan et al. algorithm, in this stage Xia and Shi form a window W of size $(2w + 1) \times (2w + 1)$ centered at the pixel (x, y) in the image $f_n(x, y)$. The flow estimate at the pixel (x, y) in this stage, denoted by (u^{k+1}, v^{k+1}), is calculated as a weighted sum of the flow vectors of the pixel within the window W.

$$\begin{aligned} u^{k+1} &= \sum_{s=-w}^{w} \sum_{t=-w}^{w} w_1\left[f_n(x,y), f_n(x+s, y+t)\right] \cdot u_c^{k+1}(x+s, y+t) \\ v^{k+1} &= \sum_{s=-w}^{w} \sum_{t=-w}^{w} w_1\left[f_n(x,y), f_n(x+s, y+t)\right] \cdot v_c^{k+1}(x+s, y+t), \end{aligned} \tag{13.49}$$

where $w_1[.,.]$ is a weight function. For each point in the window W, a weight is assigned according to the weight function. Let $(x + s, y + t)$ denote a pixel within the window W; then the weight of the pixel $(x + s, y + t)$ is given by

$$w_1\left[f_n(x,y), f_n(x+s;y+t)\right] = \frac{c}{\varepsilon + \left|f_n(x,y) - f_n(x+s, y+t)\right|}, \tag{13.50}$$

where ε is a small positive number to prevent the denominator from vanishing, c is a normalization constant that makes the summation of all the weights in the W equal 1.

From the above equation, we see that the weight is determined based on the intensity difference between the pixel under consideration and its neighboring pixel. The larger the difference in the intensity, the more likely the two points belong to different regions. Therefore, the weight will be small in this case. On the other hand, the flow vector in the same region will be similar since the corresponding weight is large. Thus, the weighting function implicitly takes flow discontinuity into account and is more advanced than that of Singh (1992) and Pan et al. (1994; 1998).

13.4.2.4 Outline of Algorithm

The following summarizes the procedures of the algorithm.

1. Perform a low-pass prefiltering on two input images to remove various noises.
2. Generate attribute images: intensity, horizontal edgeness, vertical edgeness, local contrast, and local entropy. Those attributes are computed at each grid point of both images.
3. Set the initial flow vectors to zero. Set the maximum iteration number and/or estimation accuracy.
4. For each pixel under consideration, generate flow variations according to Equation 13.44. Compute matching error for each flow variation according to Equation 13.45 and transform them to the corresponding response distribution R using Equation 13.47. Compute the flow estimation u^c, v^c using Equation 13.48.

5. Form a $(2w + 1) \times (2w + 1)$ neighborhood window W centered at the pixel. Compute the weight for each pixel within the window W using Equation 13.50. Update the flow vector using Equation 13.49.
6. Decrease the preset iteration number. If the iteration number is zero, the algorithm returns with the resultant optical flow field. Otherwise, go to the next step.
7. If the change in flow vector over two successive iterations is less than the predefined threshold, the algorithm returns with the estimated optical flow field. Otherwise, go to step 4.

13.4.2.5 Experimental Results

To compare the method with other methods existing in the literature, similar to what has been done by Pan et al. (1998) (discussed above in Section 13.3.3), the method was applied to three test sequences used by Barron et al. (1994): the "Translating Tree" sequence, the "Diverging Tree" sequence, and the "Yosemite" sequence. The same accuracy criterion is used as that by Barron et al. (1994). Only those results reported by Barron et al. (1994) with 100% density are listed in Tables 13.7, 13.8, and 13.9 for a fair and easy comparison. The Weng et al. algorithm was implemented by Xia and Shi and the results were reported by Xia and Shi (1995).

TABLE 13.7
Summary of the "Translating Tree" 2D Velocity Results

Techniques	Average Error, °	Standard Deviation, °	Density, %
Horn and Schunck (original)	38.72	27.67	100
Horn and Schunck (modified)	2.02	2.27	100
Uras et al. (unthresholded)	0.62	0.52	100
Nagel	2.44	3.06	100
Anandan	4.54	3.10	100
Singh (step 1, $n = 2$, $w = 2$)	1.64	2.44	100
Singh (step 2, $n = 2$, $w = 2$)	1.25	3.29	100
Pan, Shi, and Shu ($n = 1$, $w = 1$)	1.07	0.48	100
Weng, Ahuja, and Huang	1.81	2.03	100
Xia and Shi	0.55	0.52	100

TABLE 13.8
Summary of the "Diverging Tree" 2D Velocity Results

Techniques	Average Error, °	Standard Deviation, °	Density, %
Horn and Schunck (original)	32.43	30.28	100
Horn and Schunck (modified)	11.26	16.41	100
Uras et al. (unthresholded)	10.44	15.00	100
Nagel	11.71	10.59	100
Anandan	15.84	13.46	100
Singh (step 1, $n = 2$, $w = 2$, $N = 4$)	18.24	17.02	100
Singh (step 2, $n = 2$, $w = 2$, $N = 4$)	13.16	12.07	100
Pan, Shi, and Shu ($n = 1$, $w = 1$)	7.93	6.72	100
Weng, Ahuja, and Huang	8.41	8.22	100
Xia and Shi	7.54	6.61	100

TABLE 13.9
Summary of the "Yosemite" 2D Velocity Results

Techniques	Average Error, °	Standard Deviation, °	Density, %
Horn and Schunck (original)	12.02	11.72	100
Horn and Schunck (modified)	2.55	3.67	100
Uras et al. (unthresholded)	4.64	3.48	100
Nagel	2.94	3.23	100
Anandan (frame 19 and 21)	7.64	4.96	100
Singh (step 1, $n = 2$, $w = 2$, $N = 4$)	17.66	14.25	100
Singh (step 2, $n = 2$, $w = 2$, $N = 4$)	8.60	5.60	100
Pan, Shi, and Shu ($n = 1$, $w = 1$)	5.12	2.16	100
Weng, Ahuja, and Huang	8.01	9.71	100
Xia and Shi	4.04	3.82	100

13.4.2.6 Discussion and Conclusion

The above experimental results demonstrate that the Xia and Shi method outperforms both the Pan, Shi, and Shu method and the Weng, Ahuja, and Huang method in terms of accuracy of optical flow determined. Computationally speaking, the Xia and Shi method is less expensive than the Pan et al., since there is no correlation involved and the correlation is known to be computationally expensive.

13.5 SUMMARY

The optical flow field is a dense 2-D distribution of apparent velocities of movement of intensity patterns in image planes, while the 2-D motion field can be understood as the perspective projection of 3-D motion in the scene onto image planes. They are different. Only under certain circumstances are they equal to each other. In practice, however, they are closely related in that image sequences are usually the only data we have in motion analysis. Hence, we can only deal with the optical flow in motion analysis, instead of the 2-D motion field. The aperture problem in motion analysis refers to the problem that occurs when viewing motion via an aperture. Specifically, the only motion we can observe from local measurement is the motion component orthogonal to the underlying moving contour. That is another way to manifest the ill-posed nature of optical flow computation. In general, motion analysis from image sequences is an inverse problem, which is ill posed. Fortunately, low-level computational vision problems are only mildly ill posed. Hence, lowering the noise in image data leads to a possible significant reduction of errors in flow determination.

Numerous flow determination algorithms have appeared over the course of more than a decade. Most of the techniques take one of the following approaches: the gradient-based approach, the correlation-based approach, the energy-based approach, or the phase-based approach. In addition to these deterministic approaches, there is also a stochastic approach. A unification point of view of optical flow computation is presented in Section 13.3. That is, for any algorithm in optical flow computation, there are two types of information that need to be extracted — conservation information and neighborhood information.

Several techniques are introduced for the gradient-based approach, particularly the Horn and Schunck algorithm, which is a pioneer work in flow determination. There, the brightness invariant equation is used to extract conservation information and the smoothness constraint is used to extract neighborhood information. The modified Horn and Schunck algorithm shows significant error reduction in flow determination, because of a reduction of noise in image data, which confirms the mildly ill-posed nature of optical flow computation.

Several techniques are discussed for the correlation-based approach. The Singh algorithm is given emphasis due to its estimation-theoretical framework. The Pan, Shi, and Shu algorithm, which applies the feedback technique to the correlation method, demonstrates an accuracy enhancement in flow estimation.

Section 13.4 addresses the usage of multiple image attributes vs. that of a single image attribute in the flow determination technique. It is found that the use of multiple motion-insensitive attributes can help reduce the ambiguity in motion analysis. The application of multiple image attributes to conservation information turns out to be promising for flow computation.

Some experimental works are presented in Sections 13.3 and 13.4. With Barron et al.'s recent comprehensive survey of various existing optical flow algorithms, we can have a quantitative assessment on various optical flow techniques.

Optical flow finds application in areas such as computer vision, image interpolation, temporal filtering, and video coding. In computational vision, raising the accuracy of optical flow estimation is important. In video coding, however, lowering the bit rate for both prediction error and motion overhead, while keeping certain quality of reconstructed frames, is the ultimate goal. Properly handling the large amount of velocity vectors is a key issue in this regard. It is noted that the optical flow-based motion estimation for video compression has been applied for many years. However, the high bit overhead and computational complexity prevent it from practical usage in video coding. With the continued advance in technologies, however, we believe this problem may be resolved in the near future. In fact, an initial, successful attempt has been made and reported by Shi et al. (1998). There, based on a study that demonstrates that flow vectors are highly correlated and can be modeled by a first-order autoregressive (AR) model, the discrete cosine transform (DCT) is applied to flow vectors. An adaptive threshold technique is developed to match optical flow motion prediction and to minimize the residual errors. Consequently, this optical flow-based motion-compensated video coding algorithm achieves good performance for very low bit rate video coding. It obtains a bit rate compatible with that obtained by an H.263 standard algorithm, which uses block matching for motion estimation. (Note that the video coding standard H.263 is covered in Chapter 19.) Furthermore, the reconstructed video frames by using this flow-based algorithm are free of annoying blocking artifacts. This effect is demonstrated in Figure 13.20. Note that Figure 13.20 (b) has appeared in Figure 11.12, where the same picture is displayed in a larger size and the blocking artifacts are hence clearer.

13.6 EXERCISES

13-1. What is an optical flow field? What is a 2-D motion field? What is the difference between the two? How are they related to each other?

13-2. What is an aperture problem? Give two of your own examples.

13-3. What is the ill-posed problem? Why do we consider motion analysis from image sequences an ill-posed problem?

13-4. Is the relationship between the optical flow in an image plane and the velocities of objects in 3-D world space necessarily obvious? Justify your answer.

13-5. What does the smoothness constraint imply? Why is it required?

13-6. How are the derivatives of intensity function and the Laplacian of flow components estimated in the Horn and Schunck method?

13-7. What are the differences between the Horn and Schunck original method and the modified Horn and Schunck method? What do you observe from these differences?

13-8. What is the difference between the smoothness constraint proposed by Horn and Schunck and the oriented smoothness constraint proposed by Nagel? Provide comments.

13-9. In your own words, describe the Singh method. What is the weighted-least-square estimation technique?

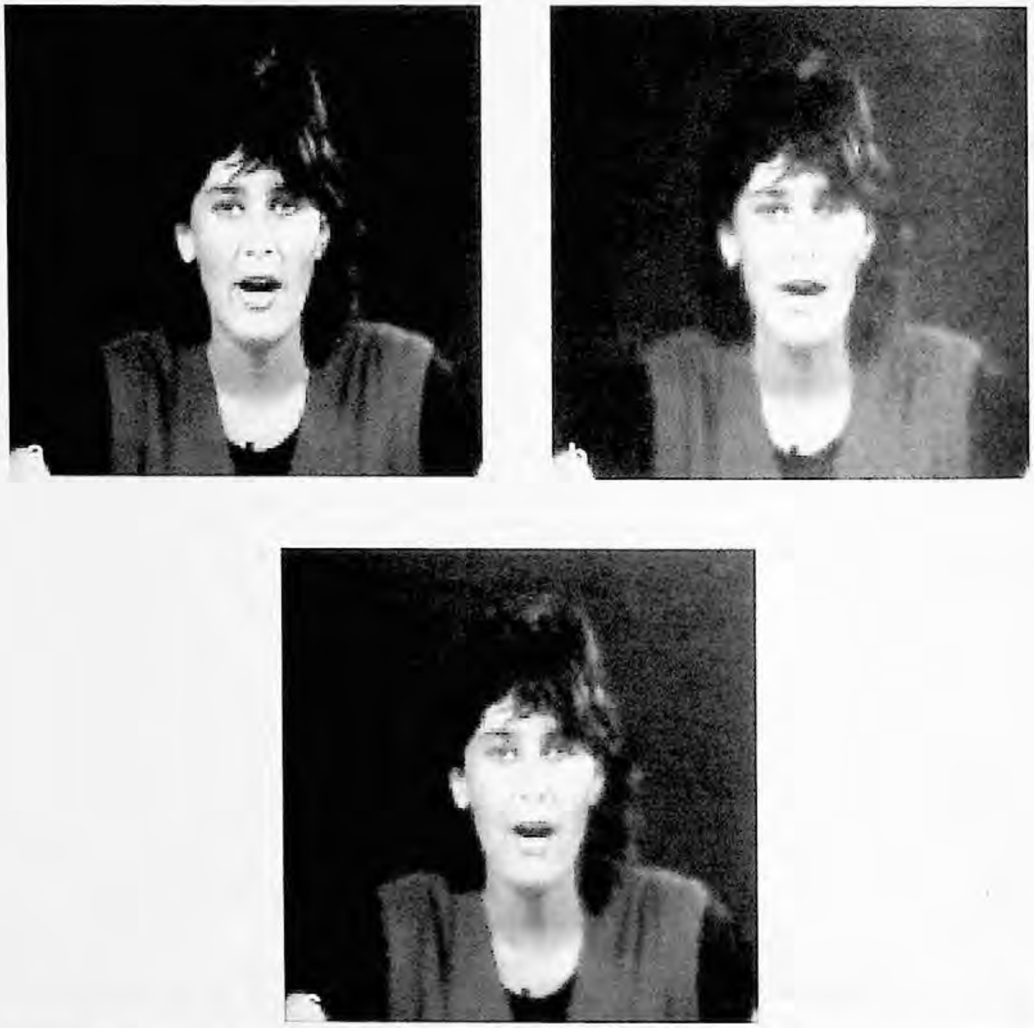


FIGURE 13.20 (a) The 21st original frame of the Miss America sequence; (b) the reconstructed 21st frame with H.263; (c) the reconstructed 21st frame with the proposed algorithm.

13-10. In your own words, describe conservation information and neighborhood information. Using this perspective, take a new look at the Horn and Schunck algorithm.

13-11. How is the feedback technique applied in the Pan et al. algorithm?

13-12. In your own words, tell the difference between the Singh method and the Pan et al. method.

13-13. Give two of your own examples to show that multiple image attributes are able to reduce ambiguity in image matching.

13-14. How does the Xia and Shi method differ from the Weng et al. method?

13-15. How does the Xia and Shi method differ from the Pan et al. method?

REFERENCES

Adelson, E. H. and J. R. Bergen, Spatiotemporal energy model for the perception of motion, *J. Opt. Soc. Am. A*, 2(2), 284-299, 1985.

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

Anandan, P. Measurement Visual Motion from Image Sequences, Ph.D. thesis, COINS Department, University of Massachusetts, Amherst, 1987.

Anandan, P. A computational framework and an algorithm for the measurement of visual motion, *Int. J. Comput. Vision*, 2, 283-310, 1989.

Barron, J. L., D. J. Fleet, and S. S. Beauchemin, Systems and experiment performance of optical flow techniques, *Int. J. Comput. Vision*, 12(1), 43-77, 1994.

Beck, J. V. and K. J. Arnold, *Parameter Estimation Engineering and Science*, John Wiley & Sons, New York, 1977.

Bertero, M., T. A. Poggio, and V. Torre, Ill-posed problems in early vision, *Proc. IEEE*, 76(8), 869-889, 1988.

Bigun, J., G. Granlund, and J. Wiklund, Multidimensional orientation estimation with applications to texture analysis and optical flow, *IEEE Trans. Pattern Anal. Machine Intell.*, 13, 775-790, 1991.

Black, M. J. and P. Anandan, The robust estimation of multiple motions: parametric and piecewise-smooth flow fields, *Comp. Vision and Image Understanding*, 63(1), 75-104, 1996.

Bracewell, R. N. *Two-Dimensional Imaging*, Prentice-Hall, Englewood Cliffs, NJ, 1995.

Burt, P. J. and E. H. Adelson, The Laplacian pyramid as a compact image code, *IEEE Trans. Commun.*, 31(4), 532-540, 1983.

Burt, P. J. The pyramid as a structure for efficient computation, in A. Rosenfeld, Ed., *Multires. Image Proc. Anal.*, 6-37, Springer Verlag, New York, 1984.

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

Fleet, D. J. and A. D. Jepson, Computation of component image velocity from local phase information, *Int. J. Comput. Vision*, 5, 77-104, 1990.

Haralick, R. M. Statistical and structural approaches to texture, *Proc. IEEE*, 67(5), 786-804, 1979.

Heeger, D. J. Optical flow using spatiotemporal filters, *Int. J. Comput. Vision*, 1, 279-302, 1988.

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

Konrad, J. and E. Dubois, Bayesian estimation of motion vector fields, *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(9), 910-927, 1992.

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

Lucas, B. and T. Kanade, An iterative image registration technique with an application to stereo vision, *Proc. DARPA Image Understanding Workshop*, 121-130, 1981.

Marr, D. *Vision*, Freeman, Boston, MA, 1982.

Nagel, H. H. Displacement vectors derived from second-order intensity variations in image sequences, *Comp. Graphics Image Proc.*, 21, 85-117, 1983.

Nagel, H. H. and W. Enkelmann, An investigation of smoothness constraints for the estimation of displacement vector fields from image sequences, *IEEE Trans. Pattern Anal. Machine Intell.*, 8, 565-593, 1986.

Pan, J. N. Motion Estimation Using Optical Flow Field, Ph.D. dissertation, Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ, 1994.

Pan, J. N., Y. Q. Shi, and C. Q. Shu, A convergence justification of the correlation-feedback algorithm in optical flow determination, Technical Report, Electronic Imaging Laboratory, Electrical and Computer Engineering Department, New Jersey Institute of Technology, Newark, NJ, 1995.

Pan, J. N., Y. Q. Shi, and C. Q. Shu, Correlation-feedback technique in optical flow determination, *IEEE Trans. Image Process.*, 7(7), 1061-1067, 1998.

Ralston, A. and P. Rabinowitz, *A First Course in Numerical Analysis*, McGraw-Hill, New York, 1978.

Sears, F. W., M. W. Zemansky, and H. D. Young, *University Physics*, Addison-Wesley, Reading, MA, 1986.

Shi, Y. Q., C. Q. Shu, and J. N. Pan, Unified optical flow field approach to motion analysis from a sequence of stereo images, *Patt. Recog.*, 27(12), 1577-1590, 1994.

Shi, Y. Q., S. Lin, and Y. Q. Zhang, Optical flow-based motion compensation algorithm for very low-bit-rate video coding, *Int. J. Imaging Syst. Technol.*, 9(4), 230-237, 1998.

Shu, C. Q. and Y. Q. Shi, Direct recovering of *N*th order surface structure using UOFF approach, *Patt. Recog.*, 26(8), 1137-1148, 1993.

Singh, A. *Optical Flow Computation: A Unified Perspective*, IEEE Computer Society Press, Los Alamitos, CA, 1991.

Singh, A. An estimation-theoretic framework for image-flow computation, *CVGIP: Image Understanding*, 56(2), 152-177, 1992.

Szeliski, R., S. B. Kang, and H.-Y. Shum, A parallel feature tracker for extended image sequences, *Proc. Int. Symp. Computer Vision*, 241-246, Florida, November 1995.

Tekalp, A. M. *Digital Video Processing*, Prentice-Hall PTR, Upper Saddle River, NJ, 1995.

Tikhonov, A. N. and V. Y. Arsenin, *Solutions of Ill-posed Problems*, Winston & Sons, Washington, D.C., 1977.

Uras, S., F. Girosi, A. Verri, and V. Torre, A computational approach to motion perception, *Biol. Cybern.*, 60, 79-97, 1988.

Waxman, A. M., J. Wu, and F. Bergholm, Convected activation profiles and receptive fields for real time measurement of short range visual motion, *Proc. IEEE Computer Vision and Pattern Recognition*, 717-723, Ann Arbor, 1988.

Weng, J., N. Ahuja, and T. S. Huang, Matching two perspective views, *IEEE Trans. PAMI*, 14(8), 806-825, 1992.

Xia, X. and Y. Q. Shi, A multiple attributes algorithm to compute optical flow, *Proc. Twenty-ninth Annual Conf. Information Sciences and Systems*, p. 480, The Johns Hopkins University, Baltimore, MD, March 1995.

Xia, X. Motion Estimation and Video Coding, Ph.D. dissertation, Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ, October, 1996.

14 Further Discussion and Summary on 2-D Motion Estimation

Since Chapter 10, we have been devoting our discussion to motion analysis and motion-compensated coding. Following a general description in Chapter 10, three major techniques — block matching, pel recursion, and optical flow — are covered in Chapters 11, 12, and 13, respectively.

In this chapter, before concluding this subject, we provide further discussion and a summary. A general characterization for 2-D motion estimation, thus for all three techniques, is given in Section 14.1. In Section 14.2, different classifications of various methods for 2-D motion analysis are given in a wider scope. Section 14.3 is concerned with a performance comparison among the three major techniques. More-advanced techniques and new trends in motion analysis and motion compensation are introduced in Section 14.4.

14.1 GENERAL CHARACTERIZATION

A few common features characterizing all three major techniques are discussed in this section.

14.1.1 Aperture Problem

The aperture problem, discussed in Chapter 13, describes phenomena that occur when observing motion through a small opening in a flat screen. That is, one can only observe normal velocity. It is essentially a form of ill-posed problem since it is concerned with existence and uniqueness issues, as illustrated in Figure 13.2(a) and (b). This problem is inherent with the optical flow technique.

We note, however, that the aperture problem also exists in block matching and pel recursive techniques. Consider an area in an image plane having strong intensity gradients. According to our discussion in Chapter 13, the aperture problem does exist in this area no matter what type of technique is applied to determine local motion. That is, motion perpendicular to the gradient cannot be determined as long as only a local measure is utilized. It is noted that, in fact, the steepest descent method of the pel recursive technique only updates the estimate along the gradient direction (Tekalp, 1995).

14.1.2 Ill-Posed Inverse Problem

In Chapter 13, when we discuss the optical flow technique, a few fundamental issues are raised. It is stated that optical flow computation from image sequences is an inverse problem, which is usually ill-posed. Specifically, there are three problems: nonexistence, nonuniqueness, and instability. That is, the solution may not exist; if it exists, it may not be unique. The solution may not be stable in the sense that a small perturbation in the image data may cause a huge error in the solution.

Now we can extend our discussion to both block matching and pel recursion. This is because both block matching and pel recursive techniques are intended for determining 2-D motion from image sequences, and are therefore inverse problems.

14.1.3 Conservation Information and Neighborhood Information

Because of the ill-posed nature of 2-D motion estimation, a unified point of view regarding various optical flow algorithms is also applicable for block matching and pel recursive techniques. That is, all three major techniques involve extracting conservation information and extracting neighborhood information.

Take a look at the block-matching technique. There, conservation information is a distribution of some sort of features (usually intensity or functions of intensity) within blocks. Neighborhood information manifests itself in that all pixels within a block share the same displacement. If the latter constraint is not imposed, block matching cannot work. One example is the following extreme case. Consider a block size of 1 × 1, i.e., a block containing only a single pixel. It is well known that there is no way to estimate the motion of a pixel whose movement is independent of all its neighbors (Horn and Schunck, 1981).

With the pel recursive technique, say, the steepest descent method, conservation information is the intensity of the pixel for which the displacement vector is to be estimated. Neighborhood information manifests itself as recursively propagating displacement estimates to neighboring pixels (spatially or temporally) as initial estimates.

In Section 12.3, it is pointed out that Netravali and Robbins suggested an alternative, called "inclusion of a neighborhood area." That is, in order to make displacement estimation more robust, they consider a small neighborhood Ω of the pixel for evaluating the square of displaced frame difference (DFD) in calculating the update term. They assume a constant displacement vector within the area. The algorithm thus becomes

$$\bar{d}^{k+1} = \bar{d}^{k} - \frac{1}{2}\alpha\nabla_{\bar{d}} \sum_{i,x,y\in Q} w_i DFD^2\left(x, y, ; \bar{d}^{k}\right), \tag{14.1}$$

where i represents an index for the ith pixel (x, y) within Ω, and w_i is the weight for the ith pixel in Ω. All the weights satisfy certain conditions; i.e., they are nonnegative, and their sum equals 1. Obviously, in this more-advanced algorithm, the conservation information is the intensity distribution within the neighborhood of the pixel, the neighborhood information is imposed more explicitly, and it is stronger than that in the steepest descent method.

14.1.4 Occlusion and Disocclusion

The problems of occlusion and disocclusion make motion estimation more difficult and hence more challenging. Here we give a brief description about these and other related concepts.

Let us consider Figure 14.1. There, the rectangle $ABCD$ represents an object in an image taken at the moment of t_{n-1}, $f(x, y, t_{n-1})$. The rectangle EFGH denotes the same object, which has been translated, in the image taken at t_n moment, $f(x, y, t_n)$. In the image $f(x, y, t_n)$, the area $BFDH$ is occluded by the object that newly moves in. On the other hand, in $f(x, y, t_n)$, the area of $AECG$ resurfaces and is referred to as a newly visible area, or a newly exposed area.

Clearly, when occlusion and disocclusion occur, all three major techniques discussed in this part will encounter a fatal problem, since conservation information may be lost, making motion estimation fail in the newly exposed areas. If image frames are taken densely enough along the temporal dimension, however, occlusion and disocclusion may not cause serious problems, since the failure in motion estimation may be restricted to some limited areas. An extra bit rate paid for the corresponding increase in encoding prediction error is another way to resolve the problem. If high quality and low bit rate are both desired, then some special measures have to be taken.

One of the techniques suitable for handling the situation is Kalman filtering, which is known as the best, by almost any reasonable criterion, technique working in the Gaussian white noise case

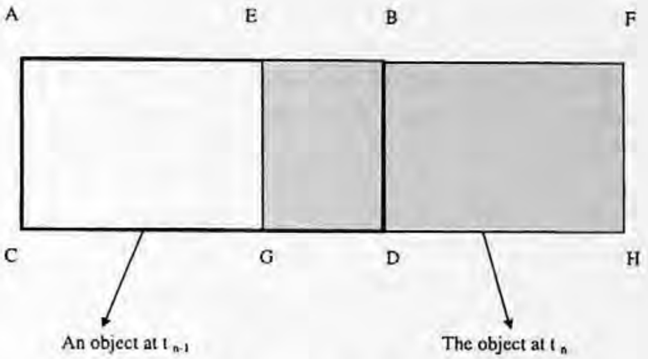


FIGURE 14.1 Occlusion and disocclusion.

(Brown and Hwang, 1992). If we consider the system that estimates the 2-D motion to be contaminated by Gaussian white noise, we can use Kalman filtering to increase the accuracy of motion estimation, particularly along motion discontinuities. It is powerful in doing incremental, dynamic, and real-time estimation.

In estimating 3-D motion, Kalman filtering was applied by Matthies et al. (1989) and Pan et al. (1994). Kalman filters were also utilized in optical flow computation (Singh, 1992; Pan and Shi, 1994). In using the Kalman filter technique, the question of how to handle the newly exposed areas was raised by Matthies et al. (1989). Pan et al. (1994) proposed one way to handle this issue, and some experimental work demonstrated its effectiveness.

14.1.5 Rigid and Nonrigid Motion

There are two types of motion: rigid motion and nonrigid motion. Rigid motion refers to motion of rigid objects. It is known that our human vision system is capable of perceiving 2-D projections of 3-D moving rigid bodies as 2-D moving rigid bodies. Most cases in computer vision are concerned with rigid motion. Perhaps this is due to the fact that most applications in computer vision fall into this category. On the other hand, rigid motion is easier to handle than nonrigid motion. This can be seen in the following discussion.

Consider a point P in 3-D world space with the coordinates (X,Y,Z), which can be represented by a column vector $\bar{v}$:

$$\bar{v} = (X,Y,Z)^T. \tag{14.2}$$

Rigid motion involves rotation and translation, and has six free motion parameters. Let R denote the rotation matrix and T the translational vector. The coordinates of point P in the 3-D world after the rigid motion are denoted by $\bar{v}'$. Then we have

$$\bar{v}' = R\bar{v} + T. \tag{14.3}$$

Nonrigid motion is more complicated. It involves deformation in addition to rotation and translation, and thus cannot be characterized by the above equation. According to the Helmholtz theory (Sommerfeld, 1950), the counterpart of the above equation becomes

$$\bar{v}' = R\bar{v} + T + D\bar{v}, \tag{14.4}$$

where D is a deformation matrix. Note that R, T, and D are pixel dependent. Handling nonrigid motion, hence, is very complicated.

In videophony and videoconferencing applications, a typical scene might be a head-and-shoulder view of a person imposed on a background. The facial expression is nonrigid in nature. Model-based facial coding has been studied extensively (Aizawa and Harashima, 1994; Li et al., 1993; Arizawa and Huang, 1995). There, a 3-D wireframe model is used for handling rigid head motion. Li (1993) analyzes the facial nonrigid motion as a weighted linear combination of a set of *action units*, instead of determining $D\bar{v}$ directly. Since the number of action units is limited, the compuatation becomes less expensive. In the Aizawa and Harashima (1989) paper, the portions in the human face with rich expression, such as lips, are *cut* and then transmitted out. At the receiving end, the portions are *pasted* back in the face.

Among the three types of techniques, block matching may be used to manage rigid motion, while pel recursive and optical flow may be used to handle either rigid or nonrigid motion.

14.2 DIFFERENT CLASSIFICATIONS

There are various methods in motion estimation, which can be classified in many different ways. We discuss some of the classifications in this section.

14.2.1 Deterministic Methods vs. Stochastic Methods

Most algorithms are deterministic in nature. To see this, let us take a look at the most prominent algorithm for each of the three major 2-D motion estimation techniques. That is, the Jain and Jain algorithm for the block matching technique (Jain and Jain, 1981); the Netravali and Robbins algorithm for the pel recursive technique (Netravali and Robbins, 1979); and the Horn and Schunck algorithm for the optical flow technique (Horn and Schunck, 1981). All are deterministic methods. There are also stochastic methods in 2-D motion estimation, such as the Konrad and Dubois algorithm (Konrad and Dubois, 1992), which estimates 2-D motion using the maximum *a posteriori* probability (MAP).

14.2.2 Spatial Domain Methods vs. Frequency Domain Methods

While most techniques in 2-D motion analysis are spatial domain methods, there are also frequency domain methods (Kughlin and Hines, 1975; Heeger, 1988; Porat and Friedlander, 1990; Girod, 1993; Kojima et al., 1993; Koc and Liu, 1998). Heeger (1988) developed a method to determine optical flow in the frequency domain, which is based on spatiotemporal filters. The basic idea and principle of the method is introduced in this subsection. A very new and effective frequency method for 2-D motion analysis (Koc and Liu, 1998) is presented in Section 14.4, where we discuss new trends in 2-D motion estimation.

14.2.2.1 Optical Flow Determination Using Gabor Energy Filters

The frequency domain method of optical flow computation developed by Heeger is suitable for highly textured image sequences. First, let us take a look at how motion can be detected in the frequency domain.

Motion in the spatiotemporal frequency domain — We initiate our discussion with a 1-D case. The spatial frequency of a (translationally) moving sinusoidal signal, ω_x, is defined as cycles per distance (usually cycles per pixel), while temporal frequency,ω_t, is defined as cycles per time unit (usually cycles per frame). Hence, the velocity of (translational) motion, defined as distance per time unit (usually pixels per frame), can be related to the spatial and temporal frequencies as follows.

$$v = \omega_t / \omega_x. \tag{14.5}$$

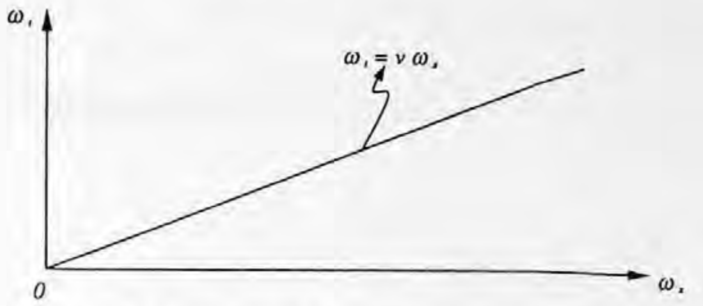


FIGURE 14.2 Velocity in 1-D spatiotemporal frequency domain.

A 1-D moving signal with a velocity v may have multiple spatial frequency components. Each spatial frequency component ω_{xi}, $i = 1,2,\ldots$ has a corresponding temporal frequency component ω_{ti} such that

$$\omega_{ti} = v\omega_{xi}. \tag{14.6}$$

This relation is shown in Figure 14.2. Thus, we see that in the spatiotemporal frequency domain, velocity is the slope of a straight line relating temporal and spatial frequencies.

For 2-D moving signals, we denote spatial frequencies by ω_x and ω_y, and velocity vector by $\vec{v} = (v_x, v_y)$. The above 1-D result can be extended in a straightforward manner as follows:

$$\omega_t = v_x\omega_x + v_y\omega_y. \tag{14.7}$$

The interpretation of Equation 14.7 is that a 2-D translating texture pattern occupies a plane in the spatiotemporal frequency domain.

Gabor Energy Filters — As Adelson and Bergen (1985) pointed out, the translational motion of image patterns is characterized by orientation in the spatiotemporal domain. This can be seen from Figure 14.3. Therefore, motion can be detected by using spatiotemporally oriented filters. One filter of this type, suggested by Heeger, is the Gabor filter.

A 1-D sine-phase Gabor filter is defined as follows:

$$g(t) = \frac{1}{\sqrt{2\pi}\sigma}\sin(2\pi\omega t)\exp\left\{-\frac{t^2}{2\sigma^2}\right\}. \tag{14.8}$$

Obviously, this is a product of a sine function and a Gaussian probability density function. In the frequency domain, this is the convolution between a pair of impulses located in ω and $-\omega$, and the Fourier transform of the Gaussian, which is itself again a Gaussian function. Hence, the Gabor function is localized in a pair of Gaussian windows in the frequency domain. This means that the Gabor filter is able to pick up some frequency components selectively.

A 3-D sine Gabor function is

$$\begin{aligned} g(x,y,t) = {} & \frac{1}{\sqrt{2}\pi^{3/2}\sigma_x\sigma_y\sigma_t}\cdot\exp\left\{-\frac{1}{2}\left(\frac{x^2}{\sigma_x^2}+\frac{y^2}{\sigma_y^2}+\frac{t^2}{\sigma_t^2}\right)\right\} \\ & \cdot\sin\left[2\pi\left(\omega_{x_0}x+\omega_{y_0}y+\omega_{t_0}t\right)\right], \end{aligned} \tag{14.9}$$

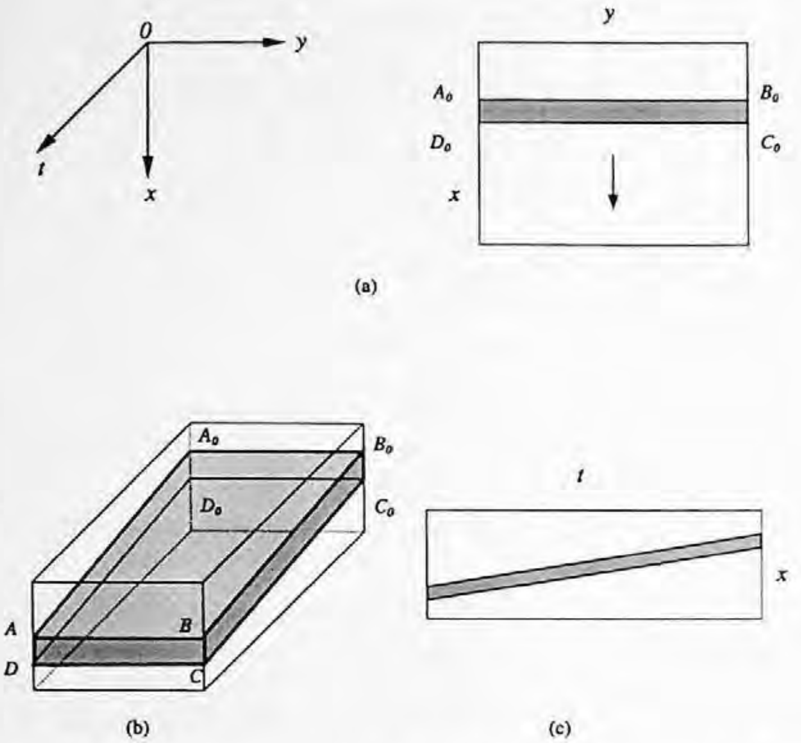


FIGURE 14.3 Orientation in spatiotemporal domain. (a) A horizontal bar translating downward. (b) A spatiotemporal cube. (c) A slice of the cube perpendicular to y axis. The orientation of the slant edges represents the motion.

where σ_x, σ_y, and σ_t are, respectively, the spreads of the Gaussian window along the spatiotemporal dimensions; and ω_{x_0}, ω_{y_0}, and ω_{t_0} are, respectively, the central spatiotemporal frequencies. The actual Gabor energy filter used by Heeger is the sum of a sine-phase filter (which is defined above), and a cosine-phase filter (which shares the same spreads and central frequencies as that in the sine-phase filter, and replaces sine by cosine in Equation 14.9). Its frequency response, therefore, is as follows.

$$\begin{aligned} G(\omega_x,\omega_y,\omega_t) = {} & \frac{1}{4}\exp\left\{-4\pi^2\left[\sigma_x^2\left(\omega_x-\omega_{x_0}\right)^2+\sigma_y^2\left(\omega_y-\omega_{y_0}\right)^2+\sigma_t^2\left(\omega_t-\omega_{t_0}\right)^2\right]\right\} \\ & +\frac{1}{4}\exp\left\{-4\pi^2\left[\sigma_x^2\left(\omega_x+\omega_{x_0}\right)^2+\sigma_y^2\left(\omega_y+\omega_{y_0}\right)^2+\sigma_t^2\left(\omega_t+\omega_{t_0}\right)^2\right]\right\}. \end{aligned} \tag{14.10}$$

This indicates that the Gabor filter is motion sensitive in that it responds largely to motion that has more power distributed near the central frequencies in the spatiotemporal frequency domain, while it responds poorly to motion that has little power near the central frequencies.

Flow extraction with motion energy — Using a vivid example, Heeger explains in his paper why one such filter is not sufficient in detection of motion. Multiple Gabor filters must be used. In fact, a set of 12 Gabor filters are utilized in Heeger's algorithm. The 12 Gabor filters in the set have one thing in common:

$$\omega_0 = \sqrt{\omega_{x0}^2 + \omega_{y0}^2}, \tag{14.11}$$

In other words, the 12 filters are tuned to the same spatial frequency band but to different spatial orientation and temporal frequencies.

Briefly speaking, optical flow is determined as follows. Denote the measured motion energy by $n_i, i = 1,2\ldots,12$. Here i indicates one of the 12 Gabor filters. The summation of all n_i is denoted by

$$\bar{n} = \sum_{i=1}^{12} n_i . \tag{14.12}$$

Denote the predicted motion energy by $P_i(v_x, v_y)$, and the sum of predicted motion energy by

$$\bar{P} = \sum_{i=1}^{12} P_i\left(v_x, v_y\right). \tag{14.13}$$

Similar to what many algorithms do, optical flow determination is then converted to a minimization problem. That is, optical flow should be able to minimize error between the measured and predicted motion energies:

$$J\left(v_x, v_y\right) = \sum_{i=1}^{12}\left[n_i - \bar{n}_i \frac{P_i\left(v_x, v_y\right)}{\bar{P}_i\left(v_x, v_y\right)}\right]^2 . \tag{14.14}$$

Similarly, many readily available numerical methods can be used for solving this minimization problem.

14.2.3 Region-Based Approaches vs. Gradient-Based Approaches

As stated in Chapter 10, methodologically speaking, there are generally two approaches to 2-D motion analysis for video coding: region based and gradient based. Now that we have gone through three major techniques, we can see this classification more clearly.

The region-based approach can be characterized as follows. For a region in an image frame, we find its best match in another image frame. The relative spatial position between these two regions produces a displacement vector. The best matching is found by minimizing a dissimilarity measure between the two regions, which is defined as

$$\sum_{(x,y)\in R}\sum M\left[f(x,y,t), f(x-dx, y-dy, t-\Delta t)\right], \tag{14.15}$$

where R denotes a spatial region, on which the displacement vector $(d_x, d_y)^T$ estimate is based; $M[\alpha,\beta]$ denotes a dissimilarity measure between two arguments α and β; Δt is the time interval between two consecutive frames.

Block matching certainly belongs to the region-based approach. By region we mean a rectangle block. For an original block in a (current) frame, block matching searches for its best match in another (previous) frame among candidates. Several dissimilarity measures are utilized, among which the mean absolute difference (MAD) is used most often.

Although it uses the spatial gradient of intensity function, the pel recursive method with inclusion of a neighborhood area assumes the same displacement vector within a neighborhood region. A weighted sum of the squared DFD within the region is used as a dissimilarity measure.

By using numerical methods such as various descent methods, the pel recursive method iteratively minimizes the dissimilarity measure, thus delivering displacement vectors. The pel recursive technique is therefore in the category of region-based approaches.

In optical flow computation, the two most frequently used techniques discussed in Chapter 13 are the gradient method and the correlation method. Clearly, the correlation method is region based. In fact, as we pointed out in Chapter 13, it is very similar to block matching.

As far as the gradient-based approach is concerned, we start its characterization with the brightness invariant equation, covered in Chapter 13. That is, we assume that brightness is conserved during the time interval between two consecutive image frames.

$$f(x,y,t) = f\left(x - d_x, y - d_y, t - \Delta t\right). \tag{14.16}$$

By expanding the right-hand side of the above equation into the Taylor series, applying the above equation, and some mathematical manipulation, we can derive the following equation.

$$f_x u + f_y v + f_t = 0, \tag{14.17}$$

where f_x, f_y, f_t are partial derivatives of intensity function with respect to x, y, and t, respectively; and u and v are two components of pixel velocity. This equation contains gradients of intensity function with respect to spatial and temporal variables and links two components of the displacement vector. The square of the left-hand side in the above equation is an error that needs to be minimized. Through the minimization, we can estimate displacement vectors.

Clearly, the gradient method in optical flow determination, discussed in Chapter 13, falls into the above framework. There, an extra constraint is imposed and included into the error represented in Equation 14.17.

Table 14.1 summarizes what we discussed in this subsection.

TABLE 14.1
Region-Based vs. Gradient-Based Approaches

			Optical Flow	
	Block Matching	Pel Recursion	Gradient-Based Method	Correlation-Based Method
Regional-based approaches	√	√		√
Gradient-based approaches			√	

14.2.4 Forward vs. Backward Motion Estimation

Motion-compensated predictive video coding may be done in two different ways: forward and backward (Boroczky, 1991). These ways are depicted in Figures 14.4 and 14.5, respectively. With the forward manner, motion estimation is carried out by using the original input video frame and the reconstructed previous input video frame. With the backward manner, motion estimation is implemented with two successive reconstructed input video frames.

The former provides relatively higher accuracy in motion estimation and hence more efficient motion compensation than the latter, owing to the fact that the original input video frames are utilized. However, the latter does not need to transmit motion vectors to the receiving end as an overhead, while the former does.

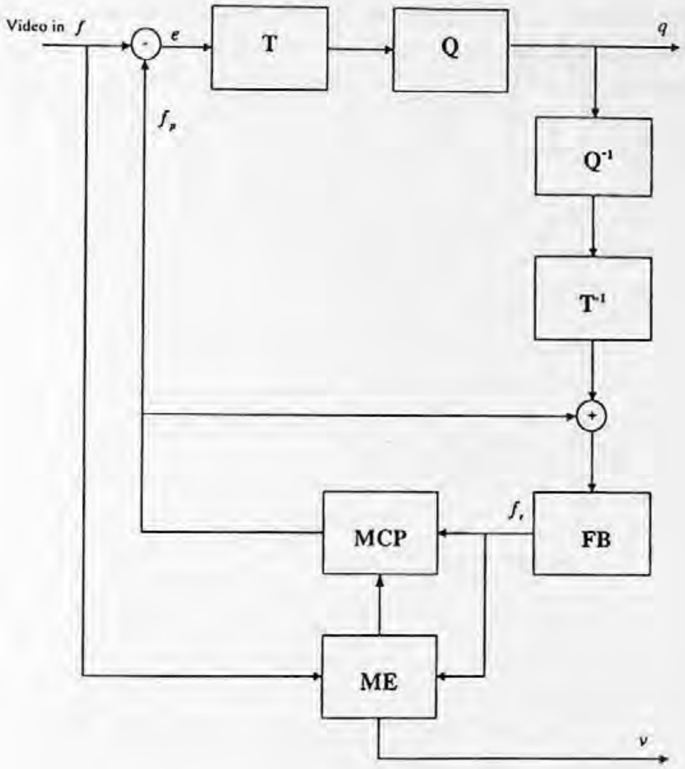


FIGURE 14.4 Forward motion estimation and compensation, T: transformer, Q: quantizer, FB: frame buffer, MCP: motion-compensated predictor, ME: motion estimator, *e*: prediction error, *f*: input video frame, f_p: predicted video frame, f_r: reconstructed video frame, *q*: quantized transform coefficients, *v*: motion vector.

Block matching is used in almost all the international video coding standards, such as H.261, H.263, MPEG 1, and MPEG 2 (which are covered in the next part of this book), as forward-motion estimation. The pel recursive technique is used as backward-motion estimation. In this way, the pel recursive technique avoids encoding a large amount of motion vectors. On the other hand, however, it provides relatively less accurate motion estimation than block matching. Optical flow is usually used as forward-motion estimation in motion-compensated video coding. Therefore, as expected, it achieves higher motion estimation accuracy on the one hand and it needs to handle a large amount of motion vectors as overhead on the other hand. These will be discussed in the next section.

It is noted that one of the new improvements in the block-matching technique is described in Section 11.6.3. It is called the predictive motion field segmentation technique (Orchard, 1993), and it is motivated by backward-motion estimation. There, segmentation is conducted *backward*, i.e., based on previously decoded frames. The purpose of this is to save overhead for shape information of motion discontinuities.

14.3 PERFORMANCE COMPARISON AMONG THREE MAJOR APPROACHES

14.3.1 Three Representatives

A performance comparison among the three major approaches; block matching, pel recursion, and optical flow, was provided in a review paper by Dufaux and Moscheni (1995). Experimental work was carried out as follows. The conventional full-search block matching is chosen as a representative

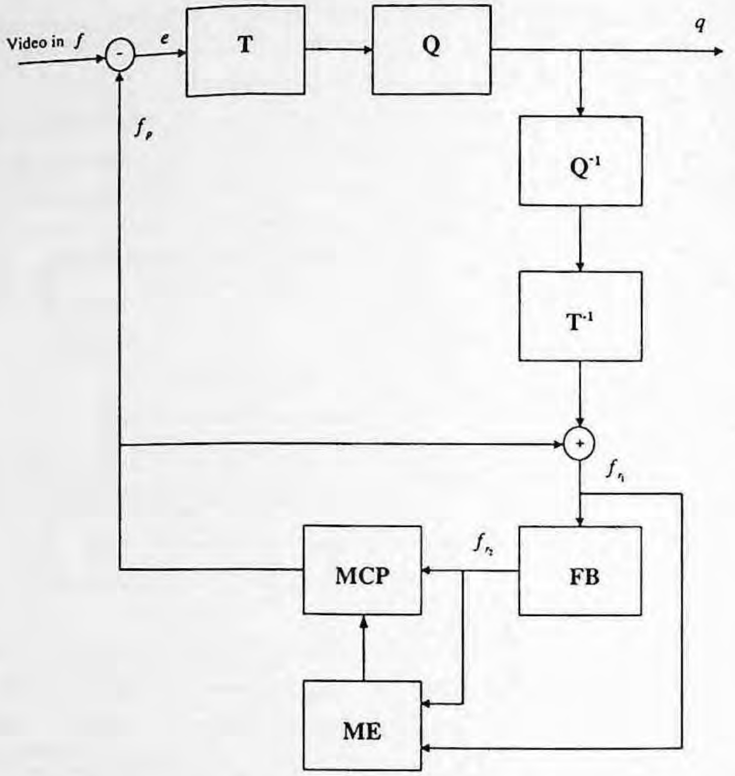


FIGURE 14.5 Backward-motion estimation and compensation, T: transformer, Q: quantizer, FB: frame buffer, MCP: motion-compensated predictor, ME: motion estimator, e: prediction error, f: input video frame, f_p: predicted video frame, f_{r1}: reconstructed video frame, f_{r2}: reconstructed previous video frame, q: quantized transform coefficients.

for the block-matching approach, while the Netravali and Robbins algorithm and the modified Horn and Schunck algorithm are chosen to represent the pel recursion and optical flow approaches, respectively.

14.3.2 Algorithm Parameters

In full-search block matching, the block size is chosen as 16 × 16 pixels, the maximum displacement is ±15 pixels, and the accuracy is half-pixel. In the Netravali and Robbins pel recursion, $\varepsilon = 1/1024$, the update term is averaged in an area of 5 × 5 pixels and clipped to a maximum of 1/16 pixels per frame, and the algorithm iterates one iteration per pixel. In the modified Horn and Schunck algorithm, the weight α^2 is set to 100, and 100 iterations of the Gauss and Seidel procedure are carried out.

14.3.3 Experimental Results and Observations

The three test video sequences are the "Mobile and Calendar," "Flower Garden," and "Table Tennis." Both subjective criteria (in terms of needle diagrams showing displacement vectors) and objective criteria (in terms of DFD error energy) are applied to access the quality of motion estimation.

It turns out that the pel recursive algorithm gives the worst accuracy in motion estimation. In particular, it cannot follow fast and large motions. Both block-matching and optical flow algorithms give better motion estimation.

It is noted that we must be cautious in drawing conclusions from these tests. This is because different algorithms in the same category and the same algorithm under different implementation conditions will provide quite different performances. In the above experiments, the full-search block matching with half-pixel accuracy is one of the better block-matching techniques. On the contrary, there are many improved pel recursive and optical flow algorithms, which outperform the chosen representatives in the reported experiments.

The experiments do, however, provide an insight about the three major approaches. Pel recursive algorithms are seldom used in video coding now, mainly because of their inaccurate motion estimation, although they do not require transmitting motion vectors to the receiving end. Although they can provide relatively accurate motion estimation, optical flow algorithms require a large amount of overhead for handling dense motion vectors. This prevents the optical flow techniques from wide and practical usage in video coding. Block matching is simple, yet very efficient for motion estimation. It provides quite accurate and reliable motion estimation for most practical video sequences in spite of its simple piecewise translational model. At the same time it does not require much overhead. Therefore, for first-generation video coding, block matching is considered to be the most suitable among the three approaches.

14.4 NEW TRENDS

In Chapters 11, 12, and 13, many new, effective improvements within the three major approaches were discussed. These techniques include multiresolution block matching, (locally adaptive) multigrid block matching, overlapped block matching, thresholding techniques, (predictive) motion field segmentation, feedback and multiple attributes in optical flow computation, subpixel accuracy, and so on. Some improvements will be discussed in Section IV, where various international video coding standards such as H.263 and MPEG 2, and 4 are introduced.

As pointed out by Orchard (1998), today our understanding of motion analysis and video compression is still based on an ad hoc framework, in general. What today's standards have achieved is not near the ideally possible performance. Therefore, more efforts are continuously made in this field, seeking much simpler and more practical, and efficient algorithms.

As an example of such developments, we conclude this chapter by presenting a novel method for 2-D motion estimation: the DCT-based motion estimation (Koc and Liu, 1998).

14.4.1 DCT-Based Motion Estimation

As pointed out in Section 14.2.2, as opposed to the conventional 2-D motion estimation techniques, this method is carried out in the frequency domain. It is also different from the Gabor energy filter method by Heeger, discussed in Section 14.2.2.1. Without introducing Gabor filters, this mehtod is directly DCT based. The fundamental concepts and techniques of this method are discussed below.

14.4.1.1 DCT and DST Pseudophases

The underlying idea behind this method is to estimate 2-D translational motion by determining the DCT and DST (discrete sine transform) *pseudophases*. Let us use the simpler 1-D case to illustrate this concept. Once it is established, it can be easily extended to the 2-D case.

Consider a 1-D signal sequence $\{f(n), n \in (0, 1, \cdots, N-1\}$ of length N. Its translated version is denoted by $\{g(n), n \in (0, 1, \cdots, N-1\}$. The translation is defined as follows.

$$g(n) = \begin{cases} f(n-d), & \text{if } (n-d) \in (0,1,\cdots,N-1) \\ 0, & \text{otherwise} \end{cases}. \qquad (14.\ 18)$$

In the above equation, d is the amount of the translation and it needs to be estimated. Let us define the following several functions before introducing the pseudophases. The DCT and the DST of the second kind of $g(n)$, $G^C(k)$, and $G^S(k)$ are defined as follows.

$$G^C(k)=\frac{2}{N}C(k)\sum_{n=0}^{N-1}g(n)\cos\left[\frac{k\pi}{N}(n+0.5)\right]\quad k\in\{0,1,\cdots N-1\} \tag{14.19}$$

$$G^S(k)=\frac{2}{N}C(k)\sum_{n=0}^{N-1}g(n)\sin\left[\frac{k\pi}{N}(n+0.5)\right]\quad k\in\{1,\cdots N\}. \tag{14.20}$$

The DCT and DST of the first kind of $f(n)$, $F^C(k)$, and $F^S(k)$ are defined as

$$F^C(k)=\frac{2}{N}C(k)\sum_{n=0}^{N-1}f(n)\cos\left[\frac{k\pi}{N}n\right]\quad k\in\{0,1,\cdots N-1\} \tag{14.21}$$

$$F^S(k)=\frac{2}{N}C(k)\sum_{n=0}^{N-1}f(n)\sin\left[\frac{k\pi}{N}n\right]\quad k\in\{1,\cdots N\}. \tag{14.22}$$

In the above equations, $C(k)$ is defined as

$$C(k)=\begin{cases}\frac{1}{\sqrt{2}} & \text{for } n=0 \text{ or } N\\ 1 & \text{otherwise}\end{cases}. \tag{14.23}$$

Now we are in a position to introduce the following equation, which relates the translational amount d to the DCT and DST of the original sequence and its translated version, defined above. That is,

$$\begin{bmatrix}G^C(k)\\ G^S(k)\end{bmatrix}=\begin{bmatrix}F^C(k) & -F^S(k)\\ F^C(k) & F^C(k)\end{bmatrix}\begin{bmatrix}D^C(k)\\ D^S(k)\end{bmatrix}, \tag{14.24}$$

where $D^C(k)$ and $D^C(k)$ are referred to as the pseudophases and defined as follows:

$$\begin{aligned}D^C(k)&\triangleq\cos\left[\frac{k\pi}{N}\left(d+\frac{1}{2}\right)\right]\\ D^S(k)&\triangleq\sin\left[\frac{k\pi}{N}\left(d+\frac{1}{2}\right)\right].\end{aligned} \tag{14.25}$$

Equation 14.24 can be solved for the amount of translation d, thus motion estimation. This becomes clearer when we rewrite the equation in a matrix-vector format. Denote the 2 × 2 matrix in Equation 14.24 by $\mathbf{F}(k)$, the 2 × 1 column vector at the left-hand side of the equation by $G(k)$, and the 2 × 1 column vector at the right-hand side by $D(k)$. It is easy to verify that the matrix $\mathbf{F}(k)$ is orthogonal by observing the following.

$$\lambda \mathbf{F}^T(k)\mathbf{F}(k) = \mathbf{I}, \tag{14.26}$$

where **I** is a 2 × 2 identity matrix and the constant λ is

$$\lambda = \frac{1}{\left[\mathbf{F}^C(k)\right]^2 + \left[\mathbf{F}^S(k)\right]^2}. \tag{14.27}$$

We then derive the matrix-vector format of Equation 14.24 as follows:

$$\vec{D}(k) = \lambda \mathbf{F}^T(k)\vec{G}(k) \quad k \in \{1, \cdots, N-1\}. \tag{14.28}$$

14.4.1.2 Sinusoidal Orthogonal Principle

It was shown above that the pseudophases, which contain the translation information, can be determined in the DCT and DST frequency domain. But how the amount of the translation can be found has not been mentioned. Here, the algorithm uses the sinusoidal principle to pick up this information. That is, the inverse DST of the second kind of scaled pseudophase, $C(k)D^s(k)$, is found to equal an algebraic sum of the following two discrete impulses according to the sinusoidal orthogonal principle:

$$ISDT\left\{C(k)D^S(k)\right\} \frac{2}{N} \sum_{k=1}^{N} C^2(k)D^S(k)\sin\left[\frac{k\pi}{N}\left(n+\frac{1}{2}\right)\right] = \delta(d-n) - \delta(d+n+1). \tag{14.29}$$

Since the inverse DST is limited to $n \in \{0, 1, \cdots, N-1\}$, the only peak value among this set of N values indicates the amount of the translation d. Furthermore, the direction of the translation (positive or negative) can be determined from the polarity (positive or negative) of the peak value.

The block diagram of the algorithm is shown in Figure 14.6. This technique can be extended to the 2-D case in a straightforward manner. Interested readers should refer to Koc and Liu (1998).

14.4.1.3 Performance Comparison

The algorithm was applied to several typical testing video sequences, such as the "Miss America" and "Flower Garden" sequences, and an "Infrared Car" sequence. The results were compared with the conventional full-search block-matching technique and several fast-search block-matching techniques such as the 2-D logarithm search, three step search, search with subsampling in the original block, and the correlation windows.

Prior to applying the algorithm, one of the following preprocessing procedures is implemented: frame differentiation or edge extraction. It was reported that for the "Flower Garden" and "Infrared Car" sequences, the DCT-based algorithm achieves a higher coding efficiency than all three fast-search block-matching methods, while for the Miss America sequence it obtains a lower efficiency. It was also reported that it performs well even in a noisy situation.

A lower computational complexity, $O(M^2)$ for an $M \times M$ search range, is one of the major advantages possessed by the DCT-based motion estimation algorithm compared with conventional full-search block matching, $O(M^2 \cdot N^2)$ for an $M \times M$ search range and an $N \times N$ block size.

With DCT-based motion estimation, a fully DCT-based motion-compensated coder structure becomes possible, which is expected to achieve a higher throughput and a lower system complexity.

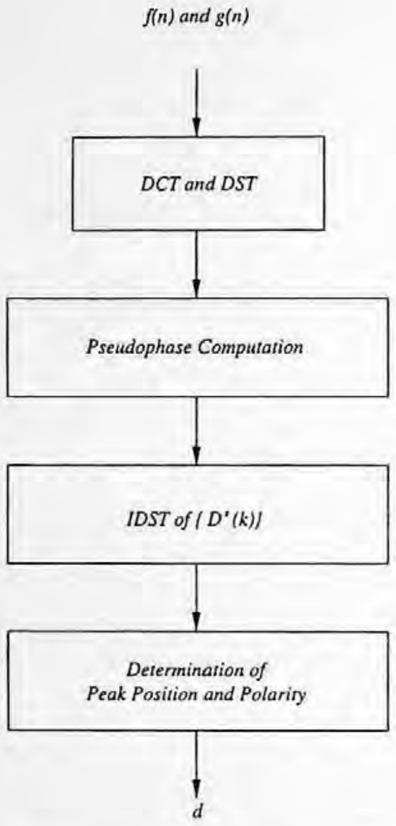


FIGURE 14.6 Block diagram of DCT-based motion estimation (1-D case).

14.5 SUMMARY

In this chapter, which concludes the motion analysis and compensation portion of the book, we first generalize the discussion of the aperture problem, the ill-posed nature, and the conservation-and-neighborhood-information unified point of view, previously made with respect to the optical flow technique in Chapter 13, to cover block-matching and pel recursive techniques. Then, occlusion and disocclusion, and rigidity and nonrigidity are discussed with respect to the three techniques. The difficulty of nonrigid motion estimation is analyzed. Its relevance in visual communications is addressed.

Different classifications of various methods in the three major 2-D motion estimation techniques; block matching, pel recursion, and optical flow, are presented. Besides the frequently utilized deterministic methods, spatial domain methods, region-based methods, and forward-motion estimation, their counterparts — stochastic methods, frequency domain methods, gradient methods, and backward motion estimation — are introduced. In particular, two frequency domain methods are presented with some detail. They are the method using the Gabor energy filter and the DCT-based method.

A performance comparison among the three techniques is also introduced in this chapter, based on which observations are drawn. A main point is that block matching is at present the most suitable technique for 2-D motion estimation among the three techniques.

14.6 EXERCISES

14-1. What is the difference between rigid motion and nonrigid motion? In facial encoding, what is the nonrigid motion? How is the nonrigid motion handled?

14-2. How is 2-D motion estimation carried out in the frequency domain? What are the underlying ideas behind the Heeger method and the Koc and Liu method?

14-3. Why is one Gabor energy filter not sufficient in motion estimation? Draw the power spectrum of a 2-D sine-phase Gabor function.

14-4. Show the correspondence of a positive (negative) peak value in the inverse DST of the second kind of DST pseudophase to a positive (negative) translation in the 1-D spatial domain.

14-5. How does neighborhood information manifest itself in the pel recursive technique?

14-6. Using your own words and some diagrams, state that the translational motion of an image pattern is characterized by orientation in the spatiotemporal domain.

REFERENCES

Adelson, E. H. and J. R. Bergen, Spatiotemporal energy models for the perception of motion, *J. Opt. Soc. Am.*, A2(2), 284-299, 1985.

Aizawa, K. and H. Harashima, Model-based analysis synthesis image coding (MBASIC) system for a person's face, *Signal Process. Image Commun.*, 139-152, 1989.

Aizawa, K. and T. S. Huang, Model-based image coding: advanced video coding techniques for very low bit rate applications, *Proc. IEEE*, 83(2), 259-271, 1995.

Boroczky, L. Pel-Recursive Motion Estimation for Image Coding, *Ph.D. dissertation*, Delft University of Technology, Netherlands, 1991.

Brown, R. G. and P. Y. C. Hwang, *Introduction to Random Signals*, 2nd ed., John Wiley & Sons, New York, 1992.

Dufaux, F. and F. Moscheni, Motion estimation techniques for digital TV: A review and a new contribution, *Proc. IEEE*, 83(6), 858-876, 1995.

Girod, B., Motion-compensating prediction with fractional-pel accuracy, *IEEE Trans. Commun.*, 41, 604, 1993.

Heeger, D. J. Optical flow using spatiotemporal filters, *Int. J. Comput. Vision*, 1, 279-302, 1988.

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

Jain, J. R. and A. K. Jain, Displacement measurement and its application in interframe image coding, *IEEE Trans. Commun.*, COM-29(12), 1799-1808, 1981.

Koc, U.-V. and K. J. R. Liu, DCT-based motion estimation, *IEEE Trans. Image Process.*, 7(7), 948-865, 1998.

Kojima, A., N. Sakurai, and J. Kishigami, Motion detection using 3D FFT spectrum, *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, V, 213-216, 1993.

Konrad, J. and E. Dubois, Bayesian estimation of motion vector fields, *IEEE Trans. Pattern Anal. Machine Intell.*, 14(9), 910-927, 1992.

Kughlin, C. D. and D. C. Hines, The phase correlation image alignment method, in *Proc. 1975 IEEE Int. Conf. on Systems, Man, and Cybernetics*, 163-165, 1975.

Li, H., P. Roivainen, and R. Forchheimer, 3-D motion estimation in model-based facial image coding, *IEEE Trans. Patt. Anal. Mach. Intell.*, 6, 545-555, 1993.

Matthies, L., T. Kanade, and R. Szeliski, Kalman filter-based algorithms for estimating depth from image sequences, *Int. J. Comput. Vision*, 3, 209-236, 1989.

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

Orchard, M. T. Predictive motion-field segmentation for image sequence coding, *IEEE Transactions Aerosp. Electron. Syst.*, 3(1), 54-69, 1993.

Orchard, M. T. Visual coding standards: a research community's midlife crisis? *IEEE Signal Processing Magazine*, 43, 1998.

Pan, J. N. and Y. Q. Shi, A Kalman filter for improving optical flow accuracy along moving boundaries, *Proceedings of SPIE 1994 Visual Communication and Image Processing*, 1, 638-649, Chicago, Sept. 1994.

Pan, J. N., Y. Q. Shi, and C. Q. Shu, A Kalman filter in motion analysis from stereo image sequences, *Proceedings of IEEE 1994 International Conference on Image Processing*, 3, 63-67, Austin, TX, Nov. 1994.

Porat, B. and B. Friedlander, A frequency domain algorithm for multiframe detection and estimation of dim targets, *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 12, 398-401, 1990.

Singh, A., Incremental estimation of image-flow using a Kalman filter, *Proc. 1991 IEEE Workshop on Visual Motion*, 36-43, Princeton, NJ, 1991.

Sommerfeld, A., *Mechanics of Deformable Bodies*, 1950.

Tekalp, A. M. *Digital Video Processing*, Prentice-Hall PTR, Upper Saddle River, NJ, 1995.

Section IV

Video Compression

15 Fundamentals of Digital Video Coding

In this chapter, we introduce the fundamentals of digital video coding which include digital video representation, rate distortion theory, and digital video formats. Also, we give a brief overview of image and video coding standards which will be discussed in the subsequent chapters.

15.1 DIGITAL VIDEO REPRESENTATION

As we discussed in previous chapters, a digital image is obtained by quantizing a continuous image both spatially and in amplitude. Digitization of the spatial coordinates is called image sampling, while digitization of the amplitude is called gray-level quantization. Suppose that a continuous image is denoted by $g(x, y)$, where the amplitude or value of g at the point (x, y) is the intensity or brightness of an image at that point. The transformation of a conntinuous image to a digital image can then be expressed as

$$f(m,n) = Q\left[g\left(x_o + m\Delta x, y_o + n\Delta y\right)\right], \tag{15.1}$$

where Q is a quantization operator, x_o and y_o are the origin of image plane, m and n are the discrete values 0, 1, 2, …, and Δx and Δy are the sampling intervals in the horizontal and vertical directions, respectively. If the sampling process is extended to a third temporal direction (or the original signal in the temporal direction is a discrete format), a sequence, $f(m,n,t)$, is obtained as introduced in Chapter 10,

$$f(m,n,t) = Q\left[g\left(x_o + m\Delta x, y_o + n\,\Delta y, t_o + t\,\Delta t\right)\right], \tag{15.2}$$

where t is the values 0, 1, 2, … and $\Delta\ t$ is the time interval.

Each point of the image or each basic element of the image is called as a pixel or pel. Each individual image is called a frame. According to the sampling theorem, the original continuous signal can be recovered exactly from its samples if the sampling frequency is higher than twice the bandwidth of the original signal (Oppenheim and Schafer, 1989). The frames are normally presented at a regular time interval so that the eye can perceive fluid motion. For example, the NTSC (National Television Systems Committee) specified a temporal sampling rate of 30 frames/second and interlace 2 to 1. Therefore, as a result of this spatio-temporal sampling, the digital signals exhibit high spatial and temporal correlation, just as the analog signals did before video data compression. In the following, we discuss the theoretical basis of video digitization. An important notion is the strong dependence between values of neighboring pixels within the same frame and between the frames themselves; this can be regarded as statistical redundancy of the image sequence. In the following section, we explain how this statistical redundancy is exploited to achieve compression of the digitized image sequence.

15.2 INFORMATION THEORY RESULTS (IV): RATE DISTORTION FUNCTION OF VIDEO SIGNAL

The principal goal in the design of a video-coding system is to reduce the transmission rate requirements of the video source subject to some picture quality constraint. There are only two ways to accomplish this goal: reduction of the statistical redundancy and psychophysical redundancy of the video source. The video source is normally very highly correlated, both spatially and temporally; that is, strong dependence can be regarded as statistical redundancy of the data source. If the video source to be coded in a transmission system is viewed by a human observer, the perceptual limitations of human vision can be used to reduce transmission requirements. Human observers are subject to perceptual limitations in amplitude, spatial resolution, and temporal acuity. By proper design of the coding system, it is possible to discard information without affecting perception, or at least, with only minimal degradation. In summary, we can use two factors: the statistical structure of the data source and the fidelity requirements of the end user, which make compression possible. The performance of the video compression algorithm depends on the several factors. First, and also fundamental, is the amount of redundancy contained in the video data source. In other words, if the original source contains a large amount of information, or high complexity, then more bits are needed to represent the compressed data. Second, if a lossy coding technique is used, by which some amount of loss is permitted in the reconstructed video data, then the performance of the coding technique depends on the compression algorithm and distortion measurements. In lossy coding, different distortion measurements will perceive the loss in different ways, giving different subjective results. The development of a distortion measure that can provide consistent numerical and subjective results is a very difficult task. Moreover, the majority of the video compression applications do not require lossless coding; i.e., it is not required that the reconstructed and original images be identical or reversible.

This intuitive explanation of how redundancy and lossy coding methods can be used to reduce source data is made more precise by the Shannon rate distortion theory (Berger, 1971), which addresses the problem of how to characterize both the source and the distortion measure. Let us consider the model of a typical visual communication system depicted in Figure 15.1. The source data is fed to the encoder system, which consists of two parts: source coding and channel coding. The function of the source coding is to remove the redundancy in both the spatial and temporal domains, whereas the function of channel coding is to insert the controlled redundancy, which is used to protect the transmitted data from the interference of channel noise. It should be noted that according to Shannon (1948) certain conditions allow the source and channel coding operations to be separated without any loss of optimality, such as when the sources are ergodic. However, Shannon did not indicate the complexity constraint on the coder involved. In practical systems that are limited

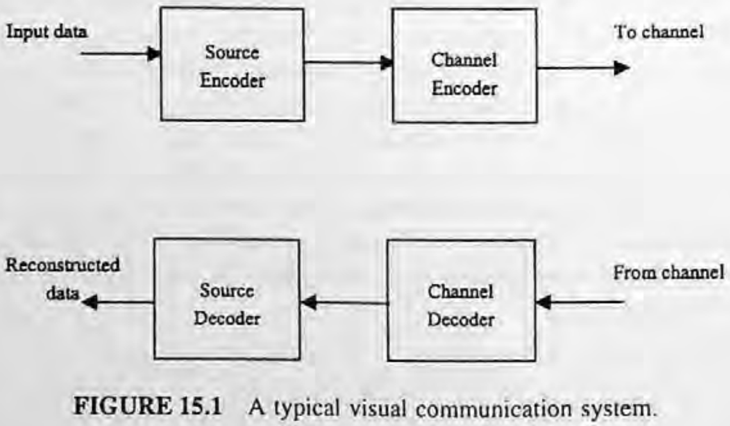


FIGURE 15.1 A typical visual communication system.

by the complexity, this separation may not be possible (Viterbi and Omura, 1979). There is still some work on the joint optimization of the source and channel coding (Modestino et al., 1981; Sayood and Borkenhagen, 1991). Returning to rate–distortion theory, the problem addressed here is the minimizing the channel capacity requirement, while maintaining the average distortion at or below an acceptable level.

The rate distortion function $R(D)$ is the minimum average rate (bits/element), and hence minimum channel capacity, required for a given average distortion level D. To make this more quantitative, we suppose that the source is a sequence of pixels, and these values are encoded by successive blocks of length N. Each block of pixels is then described by one of a denumerable set of messages, $\{X_i\}$, with probability function, $P(X_i)$. For a given input source, $\{X_i\}$, and output, $\{Y_j\}$, the decoder system can be described mathematically by the conditional probability, $Q(Y_j/X_i)$. Therefore, the probability of the output message is

$$T(Y_j) = \sum_i P(X_i)Q(Y_j/X_i). \qquad (15.3)$$

The information transmitted is called the average mutual information between Y and X and is defined for a block of length N as follows:

$$I_N(X,Y) = \sum_i \sum_j P(X_i)Q(Y_j/X_i)\log_2 \frac{Q(Y_j/X_i)}{T(Y_j)}. \qquad (15.4)$$

In the case of error-free encoding, $Y = X$ and then

$$Q(Y_j/X_i) = \begin{cases} 1, & j = i \\ 0, & j \neq i \end{cases} \quad \text{and} \quad T(Y_j) = T(Y_i). \qquad (15.5)$$

In this case, Equation 15.4 becomes

$$I_N(X,Y) = \sum_i \sum_j P(X_i)\log_2 P(X_i) = H_N(X), \qquad (15.6)$$

which is the Nth-order entropy of the data source. This can also be seen as the information contained in the data source under the assumption that no correlation exists between blocks and all the correlation between elements of each N length block is considered. Therefore, it requires at least $H_N(X)$ bits to code the data source without any information loss. In other words, the optimal error-free encoder requires $H_N(X)$ bits for the given data source. In the most general case, noise in the communication channel will result in error at least some of the time, causing $Y \neq X$. As a result,

$$I_N(X,Y) = H_N(X) - H_N(X/Y), \qquad (15.7)$$

where $H_N(X/Y)$ is the entropy of the source data at the condition of decoder output Y. Since the entropy is a positive quantity, the source entropy is the upper bound to the mutual information; i.e.,

$$I_N(X,Y) \leq H_N(X). \qquad (15.8)$$

Let d (X,Y) be the average distortion between X and Y. Then, the average distortion per pixel is defined as

$$D(Q) = \frac{1}{N} E\{d(X,Y)\} = \frac{1}{N} \sum_i \sum_j d\left(X_i, Y_j\right) P\left(X_i\right) Q\left(X_i / Y_j\right). \tag{15.9}$$

The set of all conditional probability assignments, $Q(Y/X)$, that yield average distortion less than or equal to D^*, can be written as:

$$\{Q\colon D(Q) \le D^*\}. \tag{15.10}$$

The N-block rate distortion function is then defined as the minimum of the average mutual information, $I_N(X,Y)$, per pixel:

$$R_N\left(D^*\right) = \underset{Q:D(Q)\le D^*}{\text{Min}} \frac{1}{N} I_N(X,Y). \tag{15.11}$$

The limiting value of the N-block rate distortion function is simply called the rate distortion function,

$$R\left(D^*\right) = \underset{N\to\infty}{\text{Lim}} R_N\left(D^*\right). \tag{15.12}$$

It should be clear from the above discussion that the Shannon rate distortion function is a lower bound on the transmission rate required to achieve an average distortion D when the block size is infinite. In other words, when the block size is approaching infinity, the correlation between all elements within the block is considered as the information contained in the data source. Therefore, the rate obtained is the lowest rate or lower bound. Under these conditions, the rate at which a data source produces information, subject to a requirement of perfect reconstruction, is called the entropy of the data source, i.e., the information contained in the data source. It follows that the rate distortion function is a generalization of the concept of entropy. Indeed, if the distortion measure is a perfect reproduction, it is assigned zero distortion. Then, $R(0)$ is equal to the source entropy $H(X)$. Shannon's coding theorem states that one can design a coding system with rate only negligibly greater than $R(D)$ which achieves the average distortion D. As D increases, $R(D)$ decreases monotonically and usually becomes zero at some finite value of distortion. The rate distortion function $R(D)$ specifies the minimum achievable transmission rate required to transmit a data with average distortion level D. The main value of this function in a practical application is that it potentially gives a measure for judging the performance of a coding system. However, this potential value has not been completely realized for video transmission. There are two reasons for this. First of all, there currently does not exist tractable and faithful mathematical models for an image source. The rate distortion function for Gaussian sources under the squared error distortion criterion can be found, but it is not a good model for images. The second reason is that a suitable distortion measure, D, which matches the subjective evaluation of image quality, has not been totally solved. Some results have been investigated for this task such as *JND* (just noticeable distortion) (*see* www.sarnoff.com/tech_realworld/broadcast/jnd/index.html). The issue of subjective and objective assessment of image quality has been discussed in Chapter 1. In spite of these drawbacks, the rate distortion theorem is still a mathematical basis for comparing the performance of different coding systems.

15.3 DIGITAL VIDEO FORMATS

In practical applications, most video signals are color signals. Various color systems have been discussed in Chapter 1. A color signal can be seen as a summation of light intensities of three primary wavelength bands. There are several color representations such as YC_bC_r, RGB, and others. It is common practice to convert one color representation to another color representation. The YC_bC_r color representation is used for most video coding standards in compliance with the CCIR601 (International Radio Consultative Committee), common intermediate format (CIF), and SIF formats that are described in the following. The Y component specifies the luminance information and the C_b and C_r components specify the color information. Conversion between the YC_bC_r and RGB formats can be accomplished with the following transformations, respectively.

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0.257 & 0.504 & 0.098 \\ -0.148 & -0.291 & 0.439 \\ 0.439 & -0.368 & -0.071 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix}; \tag{15.13}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.164 & 0.000 & 1.596 \\ 1.164 & -0.392 & -0.813 \\ 1.164 & 2.017 & 0.000 \end{bmatrix} \begin{bmatrix} Y-16 \\ C_b-128 \\ C_r-128 \end{bmatrix}. \tag{15.14}$$

Progressive and Interlaced — Currently, most video signals that are generated by a TV camera are interlaced. These video signals are represented at 30 frames/second for an NTSC system. Each frame consists of two fields, the top field and bottom field, which are $\frac{1}{60}$ of a second apart. In the display of an interlaced frame, the top field is scanned first and the bottom field is scanned next. The top and bottom fields are composed of alternating lines of the interlaced frame. Progressive video does not consist of fields, only frames. In an NTSC system, these frames are spaced $\frac{1}{30}$ seconds apart. In contrast to interlaced video, every line within the frame is successively scanned.

CCIR — According to CCIR601 (*see* CCIR Recommendation 601-1) (CCIR is now known as ITU-R, International Telecommunications Union-R), a color video source has three components: a luminance component (Y) and two-color difference or chrominance components (C_b and C_r or U and V in some documents). The CCIR format has two options; one for the NTSC TV system and another for the PAL TV system; both are interlaced. The NTSC format uses 525 lines/frame at 30 frames/second. The luminance frames of this format have 720 × 480 active pixels. The chrominance frames have two kinds of formats, one has 360 × 480 active pixels and is referred as the 4:2:2 format, while the other has 360 × 240 active pixels and is referred as the 4:2:0 format. The PAL format uses 625 lines/frame at 25 frames/second. Its luminance frame has 720 × 576 active pixels/frame and the chrominance frame has 360 × 576 active pixels/frame for the 4:2:2 format and 360 × 288 pixels/frame for the 4:2:0 format, both at 25 frames/second.

SIF (source input format) — SIF has luminance resolution of 360 × 240 pixels/frame at 30 frames/second or 360 × 288 pixels/frame at 25 frames/second. For both cases, the resolution of the chrominance components is half of the luminance resolution in both horizontal and vertical dimensions. SIF can easily be obtained from a CCIR format using an appropriate antialiasing filter followed by subsampling.

CIF (common intermediate format) — CIF is a noninterlaced format. Its luminance resolution has 352 × 288 pixels/frame at 30 frames/second and the chrominance has half the luminance resolution in both vertical and horizontal dimensions. Since its line value, 288, represents half the active lines in the PAL television signal, and its picture rate, 30 frames/second, is the same as the

NTSC television signal, it is a common intermediate format for both PAL or PAL-like systems and NTSC systems. In the NTSC systems, only a line number conversion is needed, while in the PAL or PAL-like systems only a picture rate conversion is needed. For low-bit-rate applications, the quarter-SIF (QSIF) or quarter-CIF (QCIF) formats may be used since these formats have only a quarter the number of pixels of SIF and CIF formats, respectively.

ATSC (Advanced Television Standard Committee) DTV (digital television) format — The concept of DTV consists of SDTV (standard-definition television) and HDTV (high-definition television). Recently, in the U.S., the FCC (Federal Communication Commission) approved the ATSC-recommended DTV standard (ATSC, 1995). The DTV format is not included in the standard due to the divergent opinions of TV and computer manufacturers. Rather, it has been agreed that the picture format will be decided by the future market. The ATSC-recommended DTV formats including two kinds of formats: SDTV and HDTV. The ATSC DTV standard includes the following 18 formats:

For HDTV: 1920 × 1080 pixels at 23.976/24 Hz, 29.97/30 Hz, and 59.94/60 Hz progressive scan.

For SDTV: 704 × 480 pixels with 4:3 aspect ratio at 23.976/24 Hz, 29.97/30 Hz, 59.94/60 Hz progressive scan; 704 × 480 pixels with 16:9 aspect ratio at 23.976/24 Hz, 29.97/30 Hz, 59.94/60 Hz progressive scan; and 640 × 480 with 4:3 aspect ratio at 23.976/24 Hz, 29.97/30 Hz, 59.94/60 Hz progressive scan.

It is noted that all HDTV formats use square pixels and only part of SDTV formats uses square pixels. The number of pixels per line vs. the number of lines/frame is known as the aspect ratio.

15.4 CURRENT STATUS OF DIGITAL VIDEO/IMAGE CODING STANDARDS

The fast growth of digital transmission services has generated a great deal of interest in the digital transmission of video signals. Since some digitized video source signals require very high bit rates, ranging from more than 100 Mbps for broadcast-quality video to more than 1 Gbps for HDTV signals, video compression algorithms which reduce the bit rates to an affordable level on practical communication channels are required. Digital video-coding techniques have been investigated over several decades. There are two factors that make video compression possible: the statistical structure of the data in the video source and the psychophysical redundancy of human vision. Video compression algorithms can remove the spatial and temporal correlation that is normally present in the video source. In addition, human observers are subject to perceptual limitations in amplitude, spatial resolution, and temporal acuity. By proper design of the coding system it is possible to discard information without affecting perceived image quality or, at least, with only minimal degradation.

Several traditional techniques have been developed for image and video data compression. Recently, with advances in data compression and VLSI (very large scale integrated) techniques, the data compression techniques have been extensively applied to video signal compression. Video compression techniques have been under development for over 20 years and have recently emerged as the core enabling technology for a new generation of DTV (both SDTV and HDTV) and multimedia applications. Digital video systems currently being implemented (or under active consideration) include terrestrial broadcasting of digital HDTV in the U.S. (ATSC, 1993), satellite DBS (Direct Broadcasting System) (Isnardi, 1993), computer multimedia (Ada, 1993), and video via packet networks (Verbiest, 1989). In response to the needs of these emerging markets for digital video, several national and worldwide standards activities have been started over the last few years. These organizations include ISO (International Standards Organization), ITU, formally known as CCITT, International Telegraph and Telephone Consultative Committee), JPEG (Joint Photographic

Experts Group), and MPEG (Motion Picture Experts Group) as shown in Table 15.1. The related standards include JPEG standards, MPEG-1,2,4 standards, and H.261 and H.263 video teleconferencing coding standards as shown in Table 15.2. It should be noted that the JPEG standards are usually used for still image coding, but they can also be used to code video. Although the coding efficiency would be lowered, they have been shown to be useful in some applications, e.g., studio editing systems. Although they are not video-coding standards and were discussed in Chapters 7 and 8, respectively, we include them here for completeness of all international image and video coding standards.

- **JPEG Standard:** Since the mid-1980s, the ITU and ISO have been working together to develop a joint international standard for the compression of still images. Officially, JPEG (ISO/IEC, 1992a) is the ISO/IEC international standard 10918-1, "Digital Compression and Coding of Continuous-Tone Still Images," or the ITU-T recommendation T.81. JPEG became an international standard in 1992. JPEG is a DCT-based coding algorithm and continues to work on future enhancements, which may adopt wavelet-based algorithms.
- **JPEG-2000:** JPEG-2000 (*see* Joint Photographic Experts Group) is a new type of image coding system under development by JPEG for still image coding. JPEG-2000 is considering using the wavelet transform as its core technique. This is because the wavelet transform can provide not only excellent coding efficiency, but also wonderful spatial and quality scalable functionality. This standard is intended to meet the need for image compression with great flexibility and efficient interchangeability. It is also intended to offer unprecedented access into the image while still in a compressed domain. Thus, an image can be accessed, manipulated, edited, transmitted, and stored in a compressed form.
- **MPEG-1:** In 1988 ISO established the MPEG to develop standards for the coded representation of moving pictures and associated audio information for digital storage applications. MPEG completed the first phase of its work in 1991. It is known as MPEG-1 (ISO/IEC, 1992b) or ISO standard 11172, "Coding of Moving Picture and Associated Audio." The target application for this specification is digital storage media at bit-rates up to about 1.5 Mbps.
- **MPEG-2:** MPEG started its second phase of work, MPEG-2 (ISO/IEC, 1994), in 1990. MPEG-2 is an extension of MPEG-1 that allows for greater input-format flexibility, higher data rate for SDTV or HDTV applications, and better error resilience. This work resulted in the ISO standard 13818 or ITU-T Recommendation H.262, "Generic Coding of Moving Pictures and Associated Audio."
- **MPEG-4:** MPEG is now working on its fourth phase, MPEG-4 (ISO/IEC, 1998). MPEG-4 visual committee draft version 1 was approved in November 1997. The end of 1999 will define the final international standard. The MPEG-4 standard supports object-based coding technology and is aimed at providing enabling technology for a variety of functionalities and multimedia applications:
  1. Universal accessibility and robustness in error-prone environments
  2. High interactive functionality
  3. Coding of natural and synthetic data or both
  4. Compression efficiency.
- **H.261:** H.261 was adopted in 1990 and the final revision was approved in 1993 by the ITU-T. It is designed for video teleconferencing and utilizes a DCT-based motion-compensation scheme. The target bit rates are from 64 to 1920 Kbps.
- **H.263, H.263 Version 2 (H.263+), H.263++ and H.26L:** The H.263 video coding standard is specifically designed for very low bit rate applications such as video conferencing. Its technical content was completed in late 1995 and the standard was approved in early 1996. It is based on the H.261 standard with several added features: unrestricted

> motion vectors, syntax-based arithmetic coding, advanced prediction, and PB-frames. The H.263 version 2 video-coding standard, also known as "H.263+," was approved in January 1998 by the ITU-T. H.263+ includes a number of new optional features based on the H.263. These new optional features are added to provide improved coding efficiency, a flexible video format, scalability, and backward-compatible supplemental enhancement information. H.263++ is the extension of H.263+ and is currently scheduled to be completed in the year 2000. H.26L is a long-term project which is looking for more efficient video-coding algorithms.

The above organizations and standards are summarized in Tables 15.1 and 15.2, respectively.

TABLE 15.1
List of Some Organizations for Standardization

Organization	Full Name of Organization
CCITT	International Telegraph and Telephone Consultative Committee
ITU	International Telecommunication Union
JPEG	Joint Photographic Experts Group
MPEG	Moving Picture Experts Group
ISO	International Standards Organization
IEC	International Electrotechnical Commission

TABLE 15.2
Video/Image Coding Standards

Name	Completion Time	Major Features
JPEG	1992	For still image coding, DCT based
JPEG-2000	2000	For still image coding, DWT based
H.261	1990	For videoconferencing, 64Kbps to 1.92 Mbps
MPEG-1	1991	For CD-ROM, 1.5 Mbps
MPEG-2 (H.262)	1994	For DTV, 2 to 15 Mbps, most extensively used
H.263	1995	For very low bit rate coding, below 64 Kbps
H.263+ (version 2)	1998	Add new optional features to H.263
MPEG-4	1999	For multimedia, content-based coding
MPEG-4 (version 2)	2000	Adds more tools to MPEG-4
H.263++	2000	Adds more optional features to H.263+
H.26L	2000	Functionally different, much more efficient
MPEG-7	2001	Content description and indexing

It should be noted that MPEG-7 in Table 15.2 is not a coding standard; it is ongoing work of MPEG. It is also interesting to note that in terms of video compression methods, there is a growing convergence toward motion-compensated, interframe DCT algorithms represented by the video coding standards. However, wavelet-based coding techniques have found recent success in the compression of still image coding in both the JPEG-2000 and MPEG-4 standards. This is because it posseses unique features in terms of high coding efficiency and excellent spatial and quality scalability. The wavelet transform has not successfully been applied to video coding due to several difficulties. For one, it is not clear how the temporal redundancy can be removed in this domain. Motion compensation is an effective technique for DCT-based video coding; however, it is not so effective for wavelet-based video coding. This is because the wavelet transform uses large block size or full frame, but motion compensation is usually performed on a limited block size. This mismatch would reduce the interframe coding efficiency. Many engineers and researchers are working on these problems.

Among these standards, MPEG-2 has had a great impact on the consumer electronics industry since the DVD (Digital Video Disk) and DTV have adopted it as core technology.

15.5 SUMMARY

In this chapter, several fundamental issues of digital video coding are presented. These include the representation and rate distortion function of digital video signals and the various video formats, which are widely used by the video industry. Finally, existing and emerging video coding standards are briefly introduced.

15.6 EXERCISES

15-1. Suppose that we have 1-D digital array (it can be extended to 2-D array that may be an image), $f(i) = X_i$, $(i = 0, 1, 2, \ldots)$. If we use the first-order linear predictor to predict the current component value with the previous component, such as: $X_i' = \alpha\ X_{i-1} + \beta$, where α and β are two parameters for this linear predictor, and if we want to minimize the mean-squared error of the prediction $\mathrm{E}\{(X_i - X_i')^2\}$, what α and β do we have to choose? Assuming that $E\{X_i\} = m$, $E\{X_i^2\} = \sigma^2$ and $E\{X_i\ X_{i-1}\} = \rho$, (for $i = 0, 1, 2, \ldots$), where m, σ, and ρ are constant.

15-2. To get a 128 × 128 or 256 × 256 digital image, write a program to use two 3 × 3 operators (Sobel operator) such as:

$$\boxed{\begin{array}{ccc} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1^{-} & 2 & 1 \end{array}} \qquad \boxed{\begin{array}{ccc} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{array}}$$

to filter the image, separately. Discuss the resulting image. What will be the result if both operators are used?

15-3. The convolution of two 2-D arrays is defined as:

$$y(m,n) = \sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} x(k,l)h(m-k,n-l)$$

and

$$\bar{x} = \begin{bmatrix} 1 & 4 & 1 \\ 2 & 5 & 3 \end{bmatrix} \quad \bar{h} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

Calculate the convolution $y(m,n)$. If $h(m,n)$ is changed to

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix},$$

recalculate $y(m,n)$.

15-4. The entropy of an image source is defined as

$$H = -\sum_{k=1}^{M} p_k \log_2 p_k ,$$

under the assumption that each pixel is an independent random variable. If the image is a binary image, i.e., $M = 2$, and the probability $p_1 + p_2 = 1$. If we define $p_1 = p$, then $p_2 = 1 - p$, $(0 \le p \le 1)$. The entropy can be rewritten as

$$H = -p \log_2 p - (1-p)\log_2(1-p).$$

Find several digital binary images and compute their entropies. If one image has almost equal number of zeros and ones and another has a different number of zeros and ones, which image has larger entropy? Prove that the entropy of a binary source is maximum if the numbers of zeros and ones are equal.

15-5. A transformation defined as $y = f(x)$, is applied to a 256×256 digital image, where x is the original pixel value and y is the transformed pixel value. Obtain new images for (a) f is a linear function, (b) f is a logarithm, and (c) f is a square function. Compare the results and indicate subjective differences of the resulting images. Repeat the experiments for different images and draw conclusions about possible use of this procedure in image processing applications.

REFERENCES

Ada, J. A. Interactive Multimedia, *IEEE Spectrum*, 22-31, 1993.

ATSC Digital Television Standard, Doc. A/53, September 16, 1995.

Berger, T. *Rate Distortion Theory — A Mathematical Bais for Data Compression*, Prentice-Hall, Englewood Cliffs, NJ, 1971.

CCIR Recommendation 601-1, Encoding parameters for digital television for studios, 1990.

Isnardi, M. Consumers seek easy to use products, *IEEE Spectrum*, 64, 1993.

ISO/IEC JTC1 IS 11172, Coding of Moving Picture and Coding of Continuous Audio for Digital Storage Media up to 1.5 Mbps, 1992b.

ISO/IEC JTC1 IS 13818, Generic Coding of Moving Pictures and Associated Audio, 1994.

ISO/IEC JTC1 FDIS 14496-2, Information Technology — Generic Coding of Audio-Visual Objects, Nov. 19, 1998.

Just Noticeable Distortion (JND) www.sarnoff.com/tech_realworld/broadcast/jnd/index.html.

Joint Photographic Experts Group (JPEG), ISO/IEC IS 11544, ITU-T Rec. T.81, 1992a.

Modestino, J. W., D. G. Daut, and A. L. Vickers, Combined source-channel coding of image using the block cosine transform, *IEEE Trans. Commun.*, COM-29, 1262-1274, 1981.

Oppenheim, A. V. and R. W. Schafer, Discrete-Time Signal Processing, Prentice-Hall, Englewood Cliffs, NJ, 1989.

Sayood, K. and J. C. Borkenhagen, Use of residual redundancy in the design of joint source/channel coders, *IEEE Trans. Commun.*, 39(6), 838-846, 1991.

Shannon, C. E. A mathematical theory of communication, *Bell Syst. Tech. J.*, 27, 379-423, 623-656, 1948.

Verbiest, W. and L. Pinnoo, A variable bit rate video codec for asynchronous transfer mode networks, *IEEE JSAC*, 7(5), 761-770, 1989.

Viterbi, A. J. and J. K. Omura, *Principles of Digital Communication and Coding*, New York: McGraw-Hill, New York, 1979.

16 Digital Video Coding Standards — MPEG-1/2 Video

In this chapter, we introduce the ISO/IEC digital video coding standards, MPEG-1 (ISO/IEC, 1992) and MPEG-2 (ISO/IEC, 1995), which are extensively used in the video industry for television broadcast, visual communications, and multimedia applications.

16.1 INTRODUCTION

As we know, MPEG has successfully developed two standards, MPEG-1 and MPEG-2. The MPEG-1 video standard was completed in 1991 with the development of the ISO/IEC specification 11172, which is the standard for coding of moving picture and associated audio for digital storage media at up to about 1.5 Mbps. To support a wide range of application profiles the user can specify a set of input parameters including flexible picture size and frame rate. MPEG-1 was developed for multimedia CD-ROM applications. Important features provided by MPEG-1 include frame-based random access of video, fast-forward/fast-reverse searches through compressed bitstreams, reverse playback of video, and editability of the compressed bitstream. MPEG-2 is formally referred to as ISO/IEC specification 13818, which is the second phase of MPEG video coding solution for applications not originally covered by the MPEG-1 standard. Specifically, MPEG-2 was developed to provide video quality not lower than NTSC/PAL and up to HDTV quality. The MPEG-2 standard was completed in 1994. Its target bit rates for NTSC/PAL are about 2 to 15 Mbps, and it is optimized at about 4 Mbps. The bit rates used for HDTV signals are about 19 Mbps. In general, MPEG-2 can be seen as a superset of the MPEG-1 coding standard and is backward compatible to the MPEG-1 standard. In other words, every MPEG-2-compatible decoder is able to decode a compliant MPEG-1 bit stream.

In this chapter, we will briefly introduce the standard itself. Since many books and publications exist for the explanation of the standards (Haskell et al., 1997; Mitchell et al., 1997), we will pay more attention to the utility of the standard, how the standard is used, and touch on some interesting research topics that have emerged. In other words, the standards provide the knowledge for how to design the decoders that are able to decode the compliant MPEG bitstreams successfully. But the standards do not specify the means of generating these bitstreams. For instance, given some bit rate, how can one generate a bitstream that provides the best picture quality? To answer this, one needs to understand the encoding process, which is an informative part of the standard (referred to as the test model), but it is very important for the content and service providers. In this chapter, the issues related to the encoding process are described. The main contents include the following topics: preprocessing, motion compensation, rate control, statistically multiplexing multiple programs, and optimal mode decision. Some of the sections contain the authors' own research results. These research results are useful in providing examples for readers to understand how the standard is used.

16.2 FEATURES OF MPEG-1/2 VIDEO CODING

It should be noted that MPEG-2 video coding has the feature of being backward compatible with MPEG-1. It turns out that most of the decoders in the market are MPEG-2 compliant decoders.

For simplicity, we will start to introduce the technical detail of MPEG-1 and then describe the enhanced features of MPEG-2, which MPEG-1 does not have.

16.2.1 MPEG-1 Features

16.2.1.1 Introduction

The algorithms employed by MPEG-1 do not provide a lossless coding scheme. However, the standard can support a variety of input formats and be applied to a wide range of applications. As we know, the main purpose of MPEG-1 video is to code moving image sequences or video signals. To achieve a high compression ratio, both intraframe redundancy and interframe redundancy should be exploited. This implies that it would not be efficient to code the video signal with an intraframe-coding scheme, such as JPEG. On the other hand, to satisfy the requirement of random access, we have to use intraframe coding from time to time. Therefore, the MPEG-1 video algorithm is mainly based on discrete cosine transform (DCT) coding and interframe motion compensation. The DCT coding is used to remove the intraframe redundancy and motion compensation is used to remove the interframe redundancy. With regard to input picture format, MPEG-1 allows progressive pictures only, but offers great flexibility in the size, up to 4095 × 4095 pixels. However, the coder itself is optimized to the extensively used video SIF picture format. The SIF is a simple derivative of the CCIR601 video format for digital television applications. According to CCIR601, a color video source has three components, a luminance component (Y) and two chrominance components (C_b and C_r) which are in the 4:2:0 subsampling format. Note that the 4:2:0 and 4:2:2 color formats were described in Chapter 15.

16.2.1.2 Layered Structure Based on Group of Pictures

The MPEG coding algorithm is a full-motion-compensated DCT and DPCM hybrid coding algorithm. In MPEG coding, the video sequence is first divided into groups of pictures or frames (GOP) as shown in Figure 16.1. Each GOP may include three types of pictures or frames: intracoded (I) picture or frame, predictive-coded (P) picture or frame, and bidirectionally predictive-coded (B) picture or frame. I-pictures are coded by intraframe techniques only, with no need for previous information. In other words, I-pictures are self-sufficient. They are used as anchors for forward and/or backward prediction. P-pictures are coded using one-directional motion-compensated prediction from a previous anchor frame, which could be either an I- or a P-picture. The distance between two nearest I-frames is denoted by N, which is the size of GOP. The distance between two nearest anchor frames is denoted by M. Parameters N and M both are user-selectable parameters, which are selected by user during the encoding. A larger number of N and M will increase the

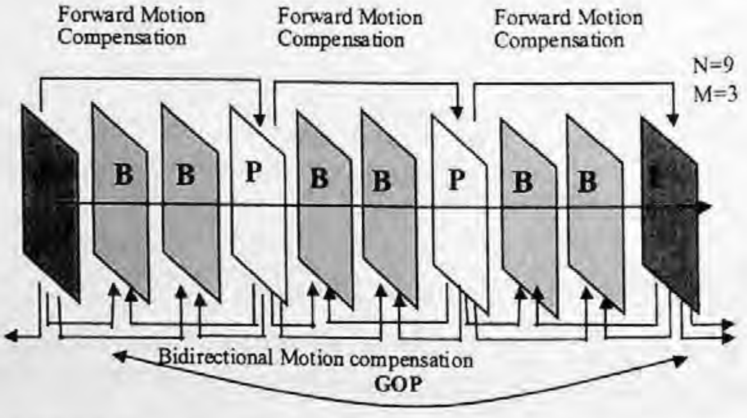

Forward Motion Compensation
Forward Motion Compensation
Forward Motion Compensation
N=9
M=3
B
B
P
B
B
P
B
B
I
Bidirectional Motion compensation
GOP

FIGURE 16.1 A group of pictures of video sequence in display order.

coding performance but cause error propagation or drift. Usually, N is chosen from 12 to 15 and M from 1 to 3. If M is selected to be 1, this means no B-picture will be used. Last, B-pictures can be coded using predictions from either past or future anchor frames (I or P), or both. Regardless of the type of frame, each frame may be divided into slices; each slice consists of several macroblocks (MBs). There is no rule to decide the slice size. A slice could contain all macroblocks in a row of a frame or all macroblocks of a frame. Smaller slice size is favorable for the purpose of error resilience, but will decrease coding performance due to higher overhead. A macroblock contains a 16×16 Y component and spatially corresponding 8×8 C_b and C_r components. A macroblock has four luminance blocks and two chrominance blocks (for 4:2:0 sampling format) and the macroblock is also the basic unit of adaptive quantization and motion compensation. Each block contains 8×8 pixels over which the DCT operation is performed.

To exploit the temporal redundancy in the video sequence, the motion vector for each macroblock is estimated from two original luminance pictures using a block-matching algorithm. The criterion for the best match between the current macroblock and a macroblock in the anchor frame is the minimum mean absolute error. Once the motion vector for each macroblock is estimated, pixel values for the target macroblock can be predicted from the previously decoded frame. All macroblocks in the I-frame are coded in intramode with no motion compensation. Macroblocks in P- and B-frames can be coded in several modes. Among the modes are intracoded and intercoded with motion compensation. This decision is made by mode selection. Most encoders depend on the values of predicted differences to make this decision. Within each slice, the values of motion vectors and DC values of each macroblock are coded using DPCM. The detailed specifications of this coding can be found in the document proposed by the MPEG video committee (ISO/IEC, 1995). The structure of MPEG implies that if an error occurs within I-frame data, it will be propagated through all frames in the GOP. Similarly, an error in a P-frame will affect the related P- and B-frames, while B-frame errors will be isolated.

16.2.1.3 Encoder Structure

The typical MPEG-1 video encoder structure is shown in Figure 16.2. Since the encoding order is different from the display order, the input sequence has to be reordered for encoding. For example, if we choose the GOP size (N) to be 12, and the distance between two nearest anchor frames (M) to be 3, the display order and encoding order are as shown in Table 16.1.

It should be noted that in the encoding order or in the bitstream the first frame in a GOP is always an I-picture. In the display order the first frame can be either an I-picture or the first B-picture of the consecutive series of B-pictures which immediately precedes the first I-picture, and the last

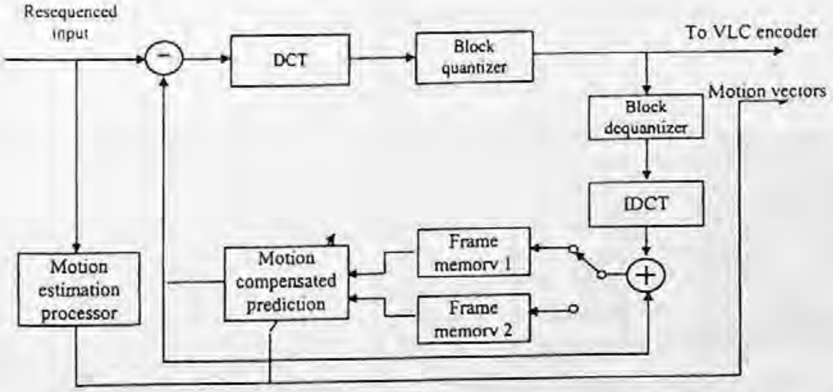


FIGURE 16.2 Typical MPEG-1 encoder structure. (From ISO/IEC, MPEG-2, Test Model 5, ISO-IEC/STCI/SC29/WGII, April, 1993. With permission.)

TABLE 16.1
Display Order and Encoding Order

Display Order	0	1	2	3	4	5	6	7	8	9	10	11	12
Encoding order	0	3	1	2	6	4	5	9	7	8	12	10	11
Coding type	I	P	B	B	P	B	B	P	B	B	I	B	B

picture in a GOP is an anchor picture, either an I- or P-picture. The first GOP always starts with an I-picture and, as a consequence, this GOP will have fewer B-pictures than the other GOPs.

The MPEG-1 video compression technique uses motion compensation to remove the interframe redundancy. The concept of motion compensation is based on the estimation of motion between video frames. The fundamental model that is used assumes that a translational motion can approximate the motion of a block. If all elements in a video scene are approximately spatially displaced, the motion between frames can be described by a limited number of motion parameters. In other words, the motion can be described by motion vectors for translatory motion of pixels. Since the spatial correlation between adjacent pixels is usually very high, it is not necessary to transmit motion information for each coded image pixel. This would be too expensive and the coder would never be able to reach a high compression ratio. The MPEG video uses the macroblock structure for motion compensation; i.e., for each 16 × 16 macroblock only one or sometimes two motion vectors are transmitted. The motion vectors for any block are found within a search window that can be up to 512 pixels in each direction. Also, the matching can be done at half-pixel accuracy, where the half-pixel values are computed by averaging the full-pixel values (Figure 16.3).

For interframe coding, the prediction differences or error images are coded and transmitted with motion information. A 2-D DCT is used for coding both the intraframe pixels and the predictive error pixels. The image to be coded is first partitioned into 8 × 8 blocks. Each 8 × 8 pixel block is then subject to an 8 × 8 DCT, resulting in a frequency domain representation of the block as shown in Figure 16.4.

The goal of the transformation is to decorrelate the block data so that the resulting transform coefficients can be coded more efficiently. The transform coefficients are then quantized. During

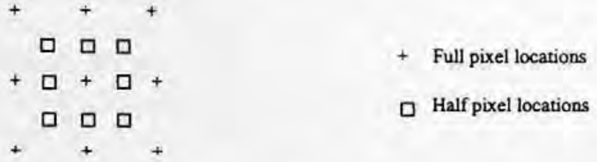
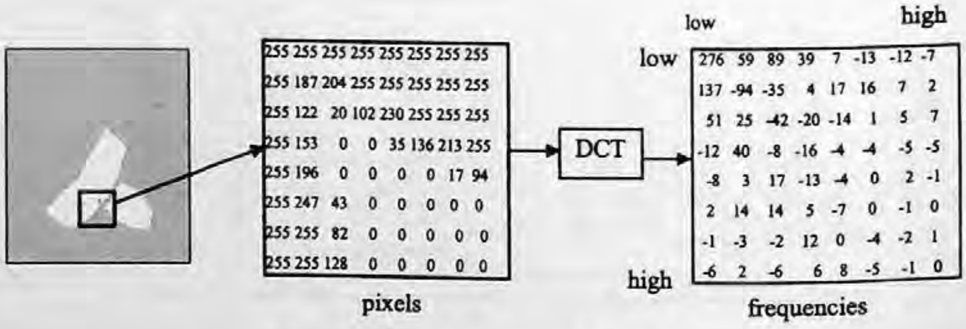


FIGURE 16.3 Half-pixel locations in motion compensation.



255	255	255	255	255	255	255	255
255	187	204	255	255	255	255	255
255	122	20	102	230	255	255	255
255	153	0	0	35	136	213	255
255	196	0	0	0	0	17	94
255	247	43	0	0	0	0	0
255	255	82	0	0	0	0	0
255	255	128	0	0	0	0	0

276	59	89	39	7	-13	-12	-7
137	-94	-35	4	17	16	7	2
51	25	-42	-20	-14	1	5	7
-12	40	-8	-16	-4	-4	-5	-5
-8	3	17	-13	-4	0	2	-1
2	14	14	5	-7	0	-1	0
-1	-3	-2	12	0	-4	-2	1
-6	2	-6	6	8	-5	-1	0

FIGURE 16.4 Example of 8 × 8 DCT.

the process of quantization a weighted quantization matrix is used. The function of quantization matrix is to quantize high frequencies with coarser quantization steps that will suppress high frequencies with no subjective degradation, thus taking advantage of human visual perception characteristics. The bits saved for coding high frequencies are used for lower frequencies to obtain better subjective coded images. There are two quantizer weighting matrices in Test Model 5 (TM5) (ISO/IEC, 1993), an intraquantizer weighting matrix and a nonintraquantizer weighting matrix; the latter is flatter since the energy of coefficients in interframe coding is more uniformly distributed than in intraframe coding.

In intra macroblocks, the DC value, *dc*, is an 11-bit value before quantization and it will be quantized to 8, 9, or 10 bits according to the setting of parameter. Thus, the quantized DC value, QDC, is calculated as

$$\text{8-bit: QDC} = dc//8,\ \text{9-bit: QDC} = dc//4,\ \text{ or 10-bit: QDC} = dc//2, \tag{16.1}$$

where symbol // means integer division with rounding to the nearest integer and the half-integer values are rounded away for zero unless otherwise specified. The AC coefficients, $ac(i, j)$, are first quantized by individual quantization factors to the value of $ac \sim (i, j)$:

$$ac \sim (i,j) = \left(16 * ac(i,j)\right) // W_I(i,j), \tag{16.2}$$

where $W_I\ (i, j)$ is the element at the (i, j) position in the intraquantizer weighting matrix shown in Figure 16.5.

The quantized level $\text{QAC}(i, j)$ is given by

$$\text{QAC}(i,j) = \left[ac \sim (i,j) + \text{sign}\left(ac \sim (i,j) * \left((p * mquant) // q\right)\right]\right/\left(2 * mquant\right), \tag{16.3}$$

where *mquant* is the quantizer scale or step which is derived for each macroblock by rate control algorithm, and $p = 3$ and $q = 4$ in TM5 (ISO/IEC, 1993). For nonintra macroblocks,

$$ac \sim (i,j) = \left(16 * ac(i,j)\right) // W_N(i,j), \tag{16.4}$$

where $W_N(i, j)$ is the nonintraquantizer weighting matrix in Figure 16.5 and

$$\text{QAC}(i,j) = ac \sim (i,j) / \left(2 * mquant\right). \tag{16.5}$$

An example of encoding an intrablock is shown in Figure 16.6.

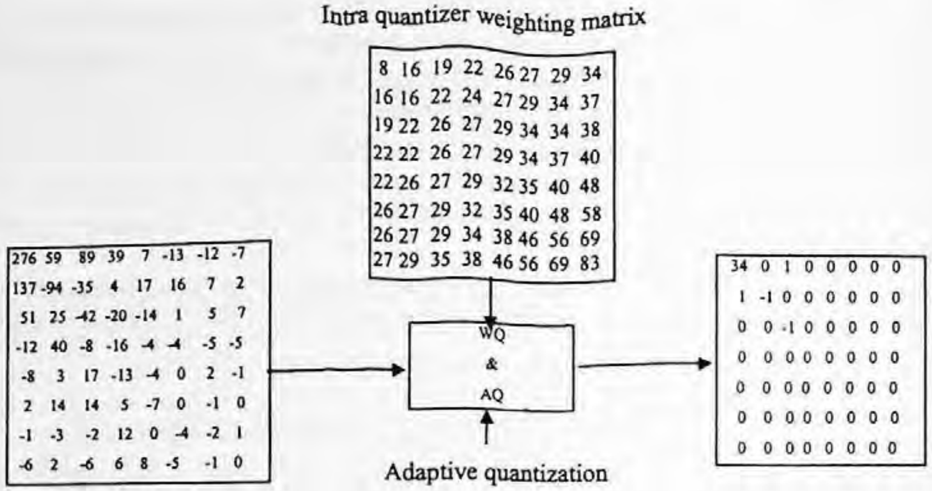
8	16	19	22	26	27	29	34
16	16	22	24	27	29	34	37
19	22	26	27	29	34	34	38
22	22	26	27	29	34	37	40
22	26	27	29	32	35	40	48
26	27	29	32	35	40	48	58
26	27	29	34	38	46	56	69
27	29	35	38	46	56	69	83

Intra quantizer weighting matrix

16	17	18	19	20	21	22	23
17	18	19	20	21	22	23	24
18	19	20	21	22	23	24	25
19	20	21	22	23	24	26	27
20	21	22	23	25	26	27	28
21	22	23	24	26	27	28	30
22	23	24	26	27	28	30	31
23	24	25	27	28	30	31	33

Nonintra quantizer weighting matrix

FIGURE 16.5 Quantizer matrices for intra- and nonintracoding.

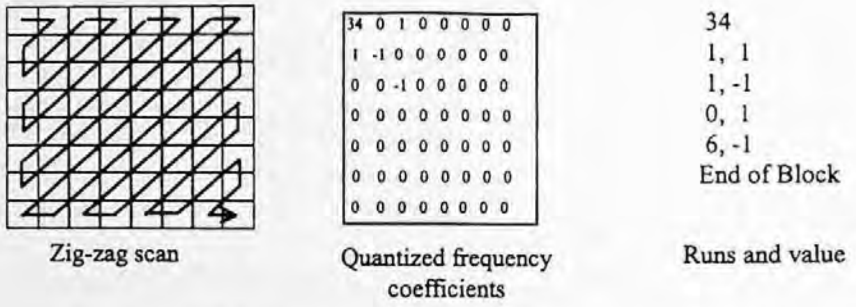


276	59	89	39	7	-13	-12	-7
137	-94	-35	4	17	16	7	2
51	25	-42	-20	-14	1	5	7
-12	40	-8	-16	-4	-4	-5	-5
-8	3	17	-13	-4	0	2	-1
2	14	14	5	-7	0	-1	0
-1	-3	-2	12	0	-4	-2	1
-6	2	-6	6	8	-5	-1	0

8	16	19	22	26	27	29	34
16	16	22	24	27	29	34	37
19	22	26	27	29	34	34	38
22	22	26	27	29	34	37	40
22	26	27	29	32	35	40	48
26	27	29	32	35	40	48	58
26	27	29	34	38	46	56	69
27	29	35	38	46	56	69	83

34	0	1	0	0	0	0	0
1	-1	0	0	0	0	0	0
0	0	-1	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

FIGURE 16.6 An example of coding an intrablock.



34	0	1	0	0	0	0	0
1	-1	0	0	0	0	0	0
0	0	-1	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

34
1, 1
1, -1
0, 1
6, -1
End of Block

FIGURE 16.7 Zigzag scans to get pairs of zero-runs and value.

The coefficients are processed in zigzag order since the most energy is usually concentrated in the lower-order coefficients. The zigzag ordering of elements in an 8 × 8 matrix allows for a more efficient run-length coder. This is illustrated in Figure 16.7.

With the zigzag order, the run-length coder converts the quantized frequency coefficients to pairs of zero runs and nonzero coefficients:

$$34\ 0\ 1\ 0\ -1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ -1\ 0\ 0\ 0\ 0 \ldots$$

After parsing we obtain the pairs of zero runs and values:

$$34\ |\ 0\ 1\ |\ 0\ -1\ |\ 1\ |\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ -1\ |\ 0\ 0\ 0\ 0 \ldots$$

These pairs of runs and values are then coded by a Huffman-type entropy coder. For example, for the above run/value, pairs are

Run/Value 34	VLC (Variable Length Code)
1, 1	0110
1, −1	0111
0, 1	110
6, −1	0001011
End of block	10

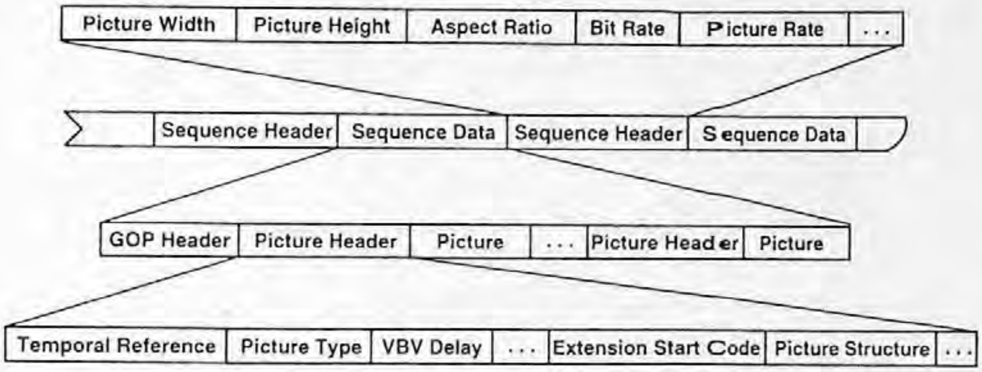


FIGURE 16.8 Description of layered structure of compressed bitstream.

The VLC tables are obtained by statistically optimizing a large number of training video sequences and are included in the MPEG-2 specification. The same idea is applied to code the DC values, motion vectors, and other information. Therefore, the MPEG video standard contains a number of VLC tables.

16.2.1.4 Structure of the Compressed Bitstream

After coding, all the information is converted to binary bits. The MPEG video bitstream consists of several well-defined layers with headers and data fields. These layers include sequence, GOP, picture, slice, macroblock, and block. The important syntax elements contained in each layer can be summarized in Table 16.2. The typical structure of the MPEG-1 video-compressed bitstream is shown in Figure 16.8. The syntax elements contained in the headers and the amount of bits defined for each element can be found in the standard.

For picture layer, a frame of picture is first partitioned into macroblocks (16 × 16 for luminance and 8 × 8 for chrominance in the 4:2:0 color representation). The compressed bitstream structure at this layer is shown in Figure 16.9. It is important to note that most elements in the syntax are coded by VLC. The tables of these variable run-length codes are obtained through the simulation of a large number of training video sequences.

TABLE 16.2
Summary of Important Syntax of Each Layer

Name of Layer	Important Syntax Elements
Sequence	Picture size and frame rate
	Bit rate and buffering requirement
	Programmable coding parameters
GOP	Random access unit
	Time code
Picture	Timing information (buffer fullness, temporal reference)
	Coding type (I, P, or B)
Slice	Intraframe addressing information
	Coding reinitialization (error resilience)
MB	Basic coding structure
	Coding mode
	Motion vectors
	Quantization
Block	DCT coefficients

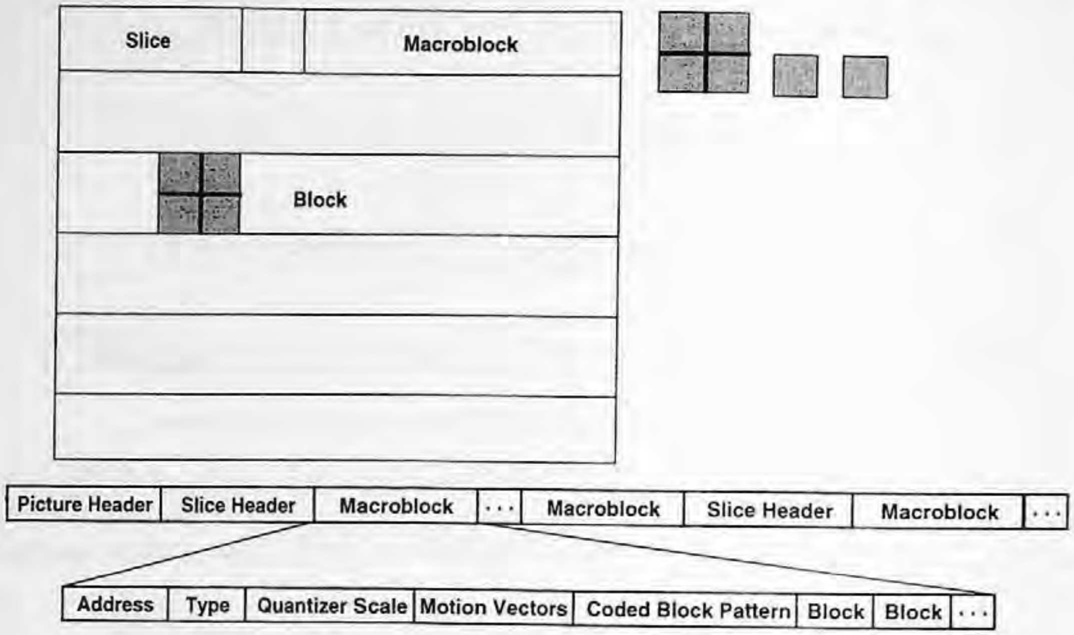


FIGURE 16.9 Picture layer data structure.

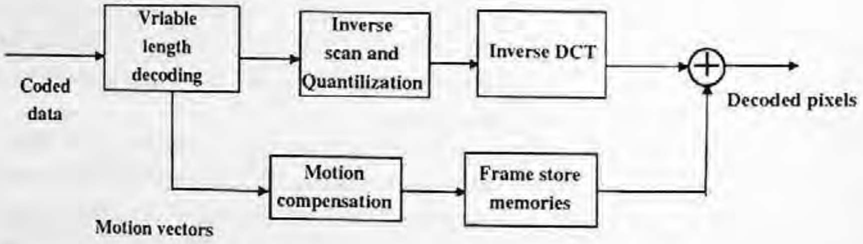


FIGURE 16.10 Simplified MPEG video decoder. (From ISO/IEC, MPEG-2 Test Model 5, April, 1993. With permission.)

16.2.1.5 Decoding Process

The decoding process is an inverse procedure of encoding. The block diagram of a typical decoder is shown in Figure 16.10

The variable-length decoder (VLD) first decodes the coded data or video bitstream. This process yields the quantized DCT coefficients and motion vector data for each macroblock. The coefficients are inversely scanned and dequantized. The decoded DCT coefficients are then inverse-transformed to obtain the spatial-domain pixels. If the macroblock was intracoded, these pixels represent the reconstructed values, without any further processing. However, if the macroblock is intercoded, then motion compensation is performed to add the prediction from the corresponding reference frame or frames.

16.2.2 MPEG-2 Enhancements

The basic coding structure of MPEG-2 video is the same as that of MPEG-1 video, that is, intraframe and interframe DCT with I-, P-, and B-pictures is used. The most important features of MPEG-2 video coding include:

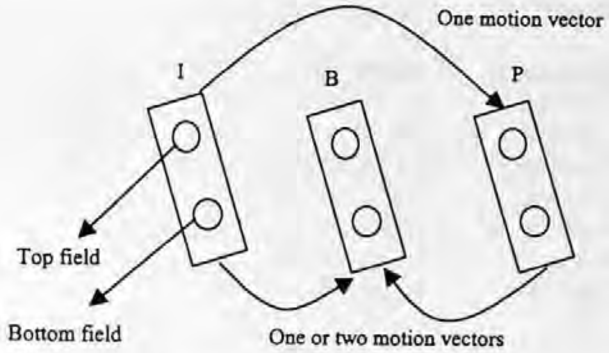


FIGURE 16.11 Frame-based prediction of MPEG-1 video coding.

- Field/frame prediction modes for supporting the interlaced video input;
- Field/frame DCT coding syntax;
- Downloadable quantization matrix and alternative scan order;
- Scalability extension.

The above enhancement items are all coding performance improvements that are related to the support of interlaced material. There are also several noncompression enhancements, which include:

- Syntax to facilitate 3:2 pull-down in the decoder;
- Pan and scan codes with $\frac{1}{16}$ pixel resolution;
- Display flags indicating chromaticity, subcarrier amplitude, and phase (for NTSC/PAL/SECAM source material).

In the following, each of these enhancements is introduced.

16.2.2.1 Field/Frame Prediction Mode

In MPEG-1 video, we always code each picture as a frame structure, whether the original material is progressive or interlaced. If the original sequence is interlaced, each frame consists of two fields: top field and bottom field as shown in Figure 16.11. We still can use frame-based prediction if we consider the two fields as a frame, such as that shown in Figure 16.11.

In Figure 16.11, three frames are coded as I-, B-, and P-frames and each frame consists of two fields. The P-frame is predicted with the I-frame with one motion vector. The B-frame can be predicted only with I-frame (forward prediction) or only with P-frame (backward prediction) or from both I- and P-picture (bidirectional prediction), the forward and backward prediction needs only one motion vector and the bidirectional prediction needs two motion vectors.

MPEG-2 video provides an enhanced prediction mode to support interlaced material, which uses the adaptive field/frame selection, based on the best match criteria. Each frame consists of two fields: top field and bottom field. Each field can be predicted from either field of the previous anchor frame. The possible prediction modes are shown in Figure 16.12.

In a field-based prediction, the top field of the current frame can be either predicted from the top field or the bottom field of an anchor frame as shown in Figure 16.12. The solid arrow represents the prediction from the top field, and the dashed arrow represents the prediction from the bottom field. The same is also true for bottom field of the current frame. If the current frame is a P-frame, there could be up to two motion vectors used to make the prediction (one for top field and one for bottom field); if the current frame is a B-frame, there could be up to four motion vectors (each field could be bidirectional prediction which needs two motion vectors). At the macroblock level

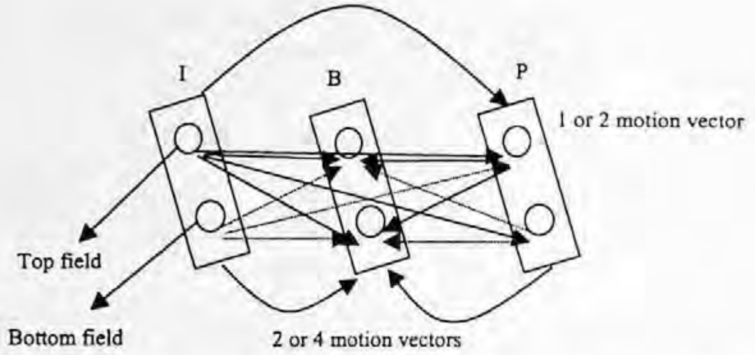


FIGURE 16.12 Field-based prediction of enhanced option of MPEG-2 video coding.

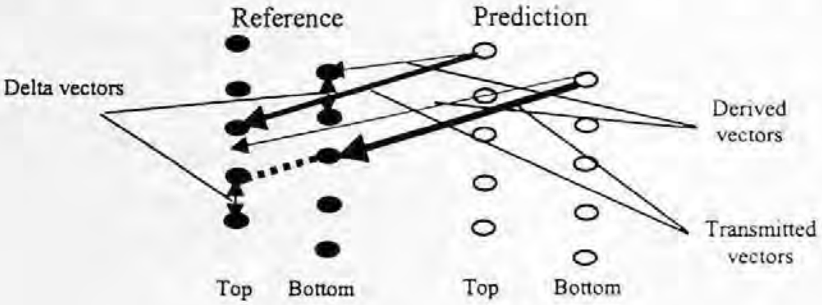


FIGURE 16.13 Dual prime prediction in MPEG-2 video coding.

of MPEG-2, several coding modes are added to support these new field-based predictions. Additionally, there is another new prediction mode supported by the MPEG-2 syntax. This is the special prediction mode referred to as dual prime prediction. The basic idea of dual prime prediction is to code a set of field motion vectors with a scaling to a near or far field, plus a transmitted delta value. Due to the correlation of adjacent pixels, the dual prime coding of field vectors can save the number of bits used for field motion vectors. The dual prime prediction is shown in Figure 16.13. In Figure 16.13, the value of one field motion vector and the value of the delta motion vector are transmitted; the motion vectors for other field are derived from the above two values.

It should be noted that only the P-picture is allowed to use dual prime prediction. In other words, if the dual prime prediction is used in the encoder, there will be no B-pictures. The reason for this restriction is to limit the required memory bandwidth for a real system implementation.

16.2.2.2 Field/Frame DCT Coding Syntax

Another important feature to support interlaced material is to allow adaptive selection of the field/frame DCT coding as shown in Figure 16.14.

In Figure 16.14, the middle is a luminance macroblock of 16 × 16 pixels, the black rectangular represents the 8 pixels in the top field and the white rectangular represents the 8 pixels in the bottom field. The left is the field DCT in which each 8 × 8 block contains only the pixels from the same field. The eight in the frame DCT, each 8 × 8 block contains the pixels from both top field and bottom field.

At the macroblock level for interlaced video, the field-type DCT may be selected when the video scene contains less detail and experiences large motion. Since the difference between adjacent fields may be large when there is large motion between fields, it may be more efficient to group the fields together, rather than the frames. In this way, the possibility that there exists more correlation among the fields can be exploited. Ultimately, this can provide much more efficient

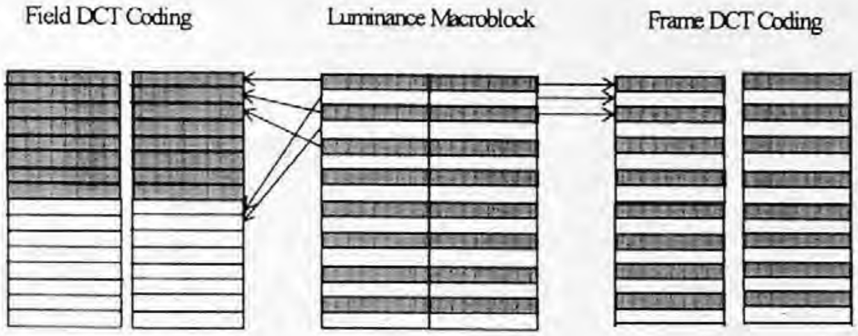


FIGURE 16.14 Frame and field DCT for interlaced video.

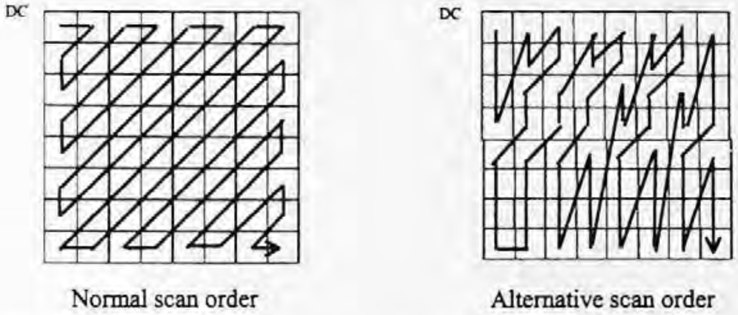


FIGURE 16.15 Two zigzag scan methods for MPEG-2 video coding.

coding since the block data are represented with fewer coefficients, especially if there is not much detail contained in the scene.

16.2.2.3 Downloadable Quantization Matrix and Alternative Scan Order

A new feature in MPEG-2 regarding the quantization matrix is that it can be downloaded for every frame. This may be helpful if the input video characteristics are very dynamic. In general, the quantizer matrices are different for intracoding and nonintracoding. With 4:2:0 format, only two matrices are used, one for the intrablocks and another for the nonintrablocks. With 4:2:2 or 4:4:4 formats four matrices are used, both an intra- and a nonintramatrix are used for the luminance and chrominance blocks. If the matrix load flags are not set, the decoder will use default matrices. The formats 4:2:0, 4:2:2 are defined in Chapter 15. In the 4:4:4 format, the luminance and two chrominance pictures have the same picture size.

In the picture layer, there is a flag that can be set for an alternative scan of DCT blocks, instead of using the zigzag scan discussed earlier. Depending on the spectral distribution, the alternative scan can yield run lengths that better exploit the multitude of zero coefficients. The zigzag scan and alternative scan are shown in Figure 16.15.

The normal zigzag scan is used for MPEG-1 and as an option for MPEG-2. The alternative scan is not supported by MPEG-1 and is an option for MPEG-2. For frame-type DCT of interlaced video, more energy may exist at the bottom part of the block; hence the run-length coding may be better off with the alternative scan.

16.2.2.4 Pan and Scan

In MPEG-2 there are several parameters defined in the sequence display extension and picture display extension. These parameters are used to display a specified rectangle within a reconstructed

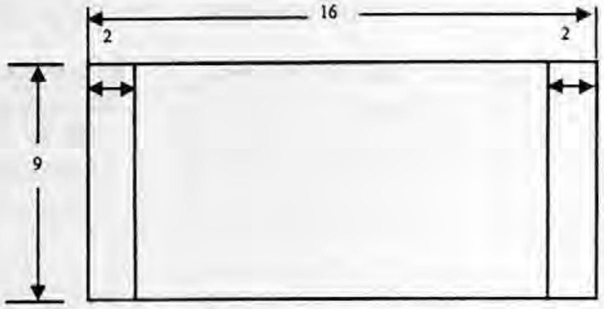


FIGURE 16.16 An example of pan-scan.

frame. They include display horizontal size and display vertical size in the sequence display extension, and frame center horizontal offset and frame center vertical offset in the picture display extension. A typical example using pan-scan parameters is the conversion of a 16:9 frame to a 4:3 frame. The 4:3 region is defined by display horizontal size and display vertical size, and the 16:9 frame is defined by horizontal size and vertical size. If we choose the display horizontal size to be 4 pixels less than the horizontal size, and keep the display vertical size as the same as the vertical size, then we can obtain a 4:3 pictures on the display. Figure 16.16 shows the conversion of 16:9 to the 4:3 frame using the pan-scan parameter, but there is no center offset involved in this example.

16.2.2.5 Concealment Motion Vector

The concealment motion vector is a new tool supported by MPEG-2. This tool is useful in concealing errors in the noisy channel environment where the transmitted data may be lost or corrupted. The basic idea of a concealment motion vector is that the motion vectors are sent for the intracoded macroblock. These motion vectors are referred to as concealment motion vectors (CMV) which should be used in macroblocks immediately below the one in which the CMV occurs. The details are described in the section about error concealment.

16.2.2.6 Scalability

MPEG-2 video has several scalable modes, which include spatial scalability, temporal scalability, SNR (signal-to-noise ratio) scalability, and data partitioning. These scalability tools allow a subset of any bitstream to be decoded into meaningful imagery. Moreover, scalability is a useful tool for error resilience on prioritized transmission media. The drawback of scalability is that some coding efficiency is lost as a result of extra overhead. Here, we briefly introduce the basic notions of the above scalability features.

Spatial scalability allows multiresolution coding, which is suitable for video service internetworking applications. In spatial scalability, a single video source is split into a base layer (lower spatial resolution) and enhancement layers (higher spatial resolution). For example, a CCIR601 video can be down-sampled to SIF format with spatial filtering, which can serve as the base layer video. The base layer or low-resolution video can be coded with MPEG-1 or MPEG-2, and the higher-resolution layer must be coded by MPEG-2-supported syntax. For the up-sampled lower layer, an additional prediction mode is available in the MPEG-2 encoder. This is a flexible technique in terms of bit rate ratios, and the enhancement layer can be used in high-quality service. The problem with spatial scalability is that there exists some bit rate penalty due to overhead and there is also a moderate increase in complexity. A block diagram that illustrates encoding with spatial scalability is shown in Figure 16.17. In Figure 16.17, the output of decoding and spatial up-sampling block provides an additional choice of prediction for the MPEG-2 compatible coder, but not the only choice of prediction. The prediction can be obtained from HDTV input itself, also depending on the prediction select criterion such as the minimum prediction difference.

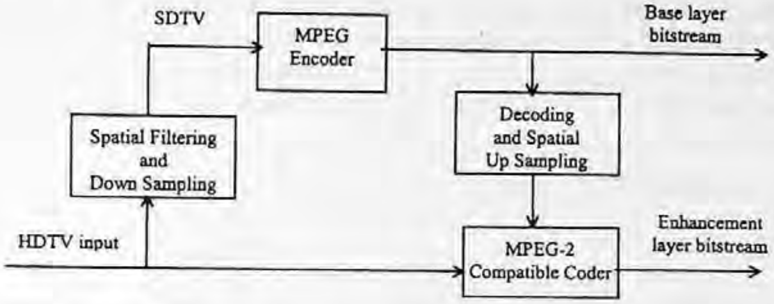

SDTV
MPEG Encoder
Base layer bitstream
Spatial Filtering and Down Sampling
Decoding and Spatial Up Sampling
HDTV input
MPEG-2 Compatible Coder
Enhancement layer bitstream

FIGURE 16.17 Block diagram of spatial scalability encoder.

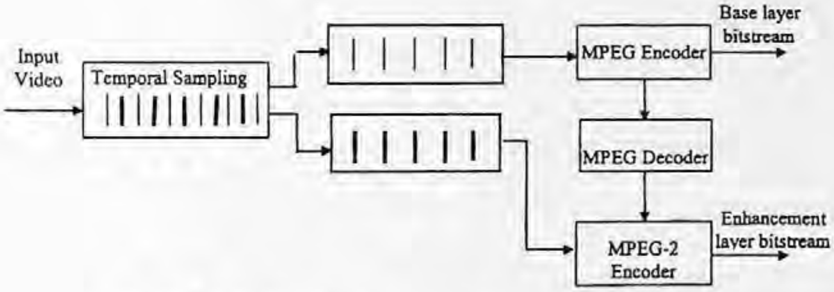


FIGURE 16.18 Block diagram of temporal scalability.

It should be noted that the spatial scalability coding allows the base layer to be coded independently from the enhancement layer. In other words, the base layer or lower layer bitstream is generated without regard for the enhancement layer and can be decoded independently. The enhancement layer bitstream is additional information, which can be seen as the prediction error based on the base layer data. This implies that the enhancement layer is useless without the base. However, this type of structure can find a lot of applications such as error concealment, which will be discussed in the following section.

Temporal scalability is a scalable coding technique in the temporal domain. An example of a two-layer temporal scalable coder is shown in Figure 16.18. The example uses temporal scalability to decompose the progressive image sequence to two interlaced image sequences; then one is coded as the base layer and one as the enhancement layer. Of course, the decomposition could be different. For the enhancement layer, there are two choices in making predictions. One choice for prediction is available between frames of base layer and enhancement layer, and the other is between frames from the enhancement layer itself. It should be noted that the spatial resolution of two layers is the same and the combined temporal rate of two layers is the full temporal rate of the source. Again, it should be noted that the decoding output of the base layer bitstream by the MPEG decoder provides an additional choice of prediction but not the only choice of predictions.

The SNR scalability provides a mechanism for transmitting two-layer service with the same spatial resolution but different quality levels. The lower layer is coded at a coarse quantization step at 3 to 5 Mbps to provide NTSC/PAL/SECAM-quality video for low-capacity channels. In the enhancement layer, the difference between original and the coarse-quantized signals is then coded with a finer quantizer to generate an enhancement bitstream for high-quality video applications.

The above three scalability schemes all generate at least two bitstreams, one for the base layer and the other for the enhancement layer, and the lower-layer bitstream can be independently decoded to provide low spatial resolution, low quality, or low frame rate video, respectively. There is another scalability scheme, data partitioning, in which the base layer bitstream cannot be independently decoded. In data partitioning, a single video source is split into a high-priority portion, which can be better protected, and low-priority portion, which is less important with regard to the reconstructed video quality. The priority breakpoint in the syntax specifies which syntax elements are coded as low priority (for example, the higher-order DCT coefficients in the intercoded blocks).

16.3 MPEG-2 VIDEO ENCODING

16.3.1 Introduction

MPEG video compression is a generic standard that is essential for the growth of the digital video industry, as mentioned previously. Although the MPEG video coding standard recommended a general coding methodology and syntax for the creation of a legitimate MPEG bitstream, there are many areas of research left open regarding how to generate high-quality MPEG bitstreams. This allows the designers of an MPEG encoder great flexibility in developing and implementing their own MPEG-specific algorithms, leading to product differentiation on the marketplace. To design a performance-optimized MPEG-2 encoder system, several major areas of research have to be considered. These include image preprocessing, motion estimation, coding mode decisions, and rate control. Algorithms for all of these areas in an encoder should aim to minimize subjective distortion for a prescribed bit rate and operating delay constraint. The preprocessing includes the noise reduction and the removal of redundant fields, which are contained in the detelecine material. The telecine material is used for the movie industry, which contains 24 progressive frames/second. The TV signal is 30 frames/second. The detelecine process converts the 24-frames/second film signal to the 30-frames/second TV signal. This is also referred to as 3:2 pull-down process. Since the 30-frames/second detelecine material only contains 24 frames/second of unique pictures, the encoder has to detect and remove the redundant fields for obtaining better coding performance. The procession of noise reduction can reduce the bits wasted for coding random noise. Motion compensation is used to remove the temporal redundancy in the video signals. The motion vectors between the anchor picture and the current picture are obtained with motion estimation algorithms. Except for I-pictures each macroblock can be inter- or intracoded, which is determined by the mode decision. The investigation of motion estimation algorithms is an important research topic since different motion estimation schemes may result in different coding efficiency. Rate control is always applied for non-variable-bit rate (non-VBR) coding. The purpose of rate control is to assign the bits for each macroblock properly under the constraints of total bit rate budget and buffer size. This is also an important topic since the optimized bit assignment scheme will result in better coding performance and better subjective reconstruct quality at a given bit rate. In this section, areas of preprocessing and motion estimation are covered. The topics of rate control and optimum mode decision are discussed in later sections.

16.3.2 Preprocessing

For low-bit-rate video coding, preprocessing is sometimes applied to the video signals before coding to increase the coding efficiency. Usually, preprocessing implies a filtering of the video signals that are corrupted by random and burst noise for various reasons, such as imperfections of the scanner, transmission, or recording medium. Noise reduction not only improves the visual quality but also increases the performance of video coding. Noise reduction can be achieved by filtering each frame independently. There are a variety of spatial filters which have been developed for image noise filtering and restoration that can be used for noise reduction task (Cano and Benard, 1983; Katsaggelos et al., 1991). On the other hand, it is also possible to filter the video sequence temporally along the motion trajectories using motion compensation (Sezan et al., 1991). However, it was shown that among the recursive stationary methods the motion-compensated spatiotemporal filtering performed better than spatial or motion-compensated temporal filtering alone (Ozkan et al., 1993).

Another important type of preprocessing is detelecine processing. Since movie material is originally shot at 24 progressive frames/second, standard conversion to television at 30 frames/second is made by a 3:2 pull-down process, which periodically inserts a repeated field, giving 30-frames/second telecine source material. The 3:2 pull-down has been described in Chapter 10, and will not be repeated here. Since the 30-frames/second detelecine material only contains 24 frames/second of unique pictures, it is necessary to detect and remove the redundant fields before or during encoding. Rather than directly encoding the 30-frames/second detelecine material, one can remove the redundant fields first and then encode 24 frames/second of unique material, thereby realizing higher coding quality at the same bit rate. The decoder can simply reconstruct the redundant fields before presenting them.

Television broadcast programmers frequently switch between telecine material and natural 30-frames/second material, such as when splicing to and from various sources of movies, ordinary television programs, and commercials. An MPEG-2 encoder should be able to cope with these transitions and consistently produce decent pictures. During movie segments, the encoder should realize the gains from coding at the lower frame rate after detelecine. Ideally, the process of source transition from the lower 24-frames/second rate to the higher 30-frames/second rate should not cause any quality drop of every encoded frame. The quality of encoded frames should maintain the same as the case where the detelecine process is ignored and all material, regardless of source type, is coded at 30 frames/second.

16.3.3 Motion Estimation and Motion Compensation

In principle, for coding video signals if the motion trajectory of each pixel could be measured, then only the initial or anchor reference frame and the motion vector information need to be coded. In such a way the interframe redundancy will be removed. To reproduce the pictures, one can simply propagate each pixel along its motion trajectory. Since there is also a cost for transmitting motion vector information, in practice one can only measure the motion vectors of a group of pixels, which will share the cost for transmission of the motion information. Of course, at the same time the pixels in the same group are assumed to have the same motion information. This is not always true since the pixels in the block may move in different directions, or some of them may belong to the background. Therefore, both motion vectors and the prediction difference have to be transmitted. Usually, the block matching can be considered as the most practical method for motion estimation because of less hardware complexity. In the block-matching method, the image frame is divided into fixed-size small rectangular blocks such as 16 × 16 or 16 × 8 in MPEG video coding. Each block is assumed to undergo a linear translation and the displacement vector of each block and the predictive errors are coded and transmitted. The related issues for motion estimation and compensation include a motion vector searching algorithm, searching range, matching criteria and coding method. Although the matching criteria, and searching algorithms have been discussed in Chapter 11, we will briefly introduce them here for the sake of completeness.

16.3.3.1 Matching Criterion

The matching of the blocks can be determined according to the various criteria including the maximum cross-correlation, the minimum mean square error (MSE), the minimum mean absolute difference (MAD) and maximum matching pixel count (MPC). For MSE and MAD, the best matching block is reached if the MSE or MAD is minimized at that location. In practice, we use MAD instead of MSE as the matching criterion because of its computational simplicity. The minimum MSE criterion is not commonly used in hardware implementations because it is difficult to realize the square operation. However, the performance of the MAD criterion deteriorates as the search area becomes larger as a result of the presence of several local minima. In the maximum MPC criterion, each pixel in the block is classified as either a matching pixel or a mismatching pixel according to the prediction difference whether which is smaller than a preset threshold. The best matching is then determined by the maximum number of matching pixels. However, the MPC criterion requires a threshold comparator and a counter.

16.3.3.2 Searching Algorithm

Finding the best-matching block requires optimizing the matching criterion over all possible candidate displacement vectors at each pixel. The so-called full-search logarithmic search, and hierarchical searching algorithms can accomplish this.

Full search: The full-search algorithm evaluates the matching criterion for all possible values within the predefined searching window. If the search window is restricted to a $[-p, p]$ square, for each motion vector there are $(2p + 1)^2$ search locations. For a block size of $M \times N$ pixels, at each search location we compare $N \times M$ pixels. If we know the matching criterion and how many operations are needed for each comparison, then we can calculate the computation complexity of the full-search algorithm. Full search is computationally expensive, but guarantees finding the global optimal matching within a defined searching range.

Logarithmic search: Actually, the expected accuracy of motion estimation algorithms varies according to the applications. In motion-compensated video coding, all one seeks is a matching block in terms of some metric, even if the match does not correlate well with the actual projected motion. Therefore, in most cases, search strategies faster than full searches are used, although they lead to suboptimal solutions. These faster search algorithms evaluate the criterion function only at a predetermined subset of the candidate motion vector locations instead of all possible locations. One of these faster search algorithms is the logarithmic search. Its more popular form is referred to as the three-step search. We explain the three-step search algorithm with the help of Figure 16.19, where only the search frame is depicted. Search locations corresponding to each of the steps in the three-step search procedure are labeled 1, 2, and 3. In the first step, starting from pixel 0 we compute MAD for the nine search locations labeled 1. The spacing between these search locations here is 4. Assume that MAD is minimum for the search location (4,4) which is circled 1. In the second step, the criterion function is evaluated at eight locations around the circled 1 which are labeled 2. The spacing between locations is now 2 pixels. Assume now the minimum MAD is at the location (6,2), which is also circled. Thus, the new search origin is the circled 2, which is located at (6.2). For the third step, the spacing is now set to 1 and the eight locations labeled 3 are searched. The search procedure is terminated at this point and the output of the motion vector is (7,1). Additional steps may be incorporated into the procedure if we wish to obtain subpixel accuracy in the motion estimations. Then, the search frame needs to be interpolated to evaluate the criterion function at subpixel locations.

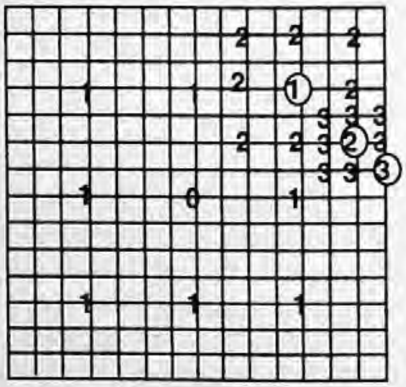


FIGURE 16.19 Three-step search.

Hierarchical motion estimation: Hierarchical representations of images in the form of a Laplacian pyramid or wavelet transform are also quite often used with the block-matching method for improved motion estimation. The basic idea of hierarchical block matching is to perform motion estimation at each level successively, starting with the lowest resolution level. The lower resolution levels serve to determine a rough estimate of the motion information using relatively larger blocks. The estimate of the motion vector at a lower resolution level is then passed onto the next higher resolution level as an initial estimate. The higher resolution levels are used to fine-tune the motion vector estimate. At higher resolution levels, relatively smaller window sizes can be used since we start with a good initial estimate. The hierarchical motion estimate can significantly reduce the implementation complexity since its search method is very efficient. However, such a method requires increased storage because of the need to keep pictures at different resolutions. Furthermore, this scheme may yield inaccurate motion vectors for regions containing small objects. Since the search starts at the lowest resolution of the hierarchy, regions containing small objects may be eliminated and thus fail to be tracked. On the other hand, the creation of low-resolution pictures provides some immunity to noise. Results of experiments performed by one of the authors have shown that, compared with full-search, the two-layer hierarchical motion estimation reduces the search complexity of factor 10 at the price of degrading reconstruction quality from about 0.2 to 0.6 dB for frame-mode coding, from 0.26 to 0.38 dB for field-mode coding, and only 0.16 to 0.37 dB for frame/field adaptive coding, for different video sequences in the case of a fixed bit rate of 4 Mbps. In the case of VBR coding, similar results can be observed from the rate distortion curves.

In the above discussion, we have restricted the motion vector estimation to integer pixel grids, or pixel accuracy. Actually, the motion vectors can be estimated with fractional or subpixel accuracy. In MPEG-2 video coding the half-pixel accuracy motion estimation can be used. Half-pixel accuracy can easily be achieved by interpolating the current and reference pictures by a factor of two and then using any of the motion estimation methods described previously.

16.3.3.3 Advanced Motion Estimation

Progress has recently been made in several aspects of motion estimation, which are described as follows.

Motion estimation using a reduced set of image data: The methods to reduce search complexity with subsampling and pyramid processing are well known and can be found in the literatures (Sun, 1994). However, the reduction by lowering the precision of each sample does not appear to have been extensively studied. Some experimental results have shown that performance degradation of the hierarchical motion estimation algorithm is not serious when each layer up to a four-layer pyramid is limited to 6 bits/sample. At 4 to 5 bits/sample the performance is degraded 0.2 dB over full precision.

Overlapped motion estimation (Katto et al., 1994): A limitation of block matching is that it generates a significant proportion of motion vectors that do not represent the true motion present in the scene. One possible reason is that the motion vectors are estimated without reference to any picture data outside of the nonoverlapping blocks. This problem has been addressed by overlapped motion estimation. In the case of the overlapped motion compensation, motion-compensated regions translated by the motion vectors are overlapped with each other. Then, a window function is used to determine the weighting factors for each vector. This technique has been adopted into the H.263 video coding standard. Some improvements have been clearly identified for low-bit-rate coding.

Frequency domain motion estimation: An alternative to spatial-domain block-matching methods is to estimate motion vectors in the frequency domain through calculating the cross-correlation (Young and Kingsbury, 1993). Most international standards, such as MPEG, H.263, as well as the proposed HDTV standard, use the DCT and block-based motion estimation as essential elements to achieve spatial and temporal compression, respectively. The new motion estimation approach is proposed in the DCT domain (Koc and Liu, 1998). This method of motion estimation has certain merits over conventional methods. It has very low computational complexity and is robust even in a noisy environment. Moreover, the motion-compensation loop in the encoder is much simplified due to replacing the IDCT out of the loop (Koc and Liu, 1998).

Generalized block matching: In generalized block matching, the encoded frame is divided into triangular, rectangular, or arbitrary quadrilateral patches. We then search for the best-matching triangular or quadrilateral patch in the search frame under a given spatial transformation. The choice of patch shape and the spatial transform are mutual related. For example, triangular patches offer sufficient degrees of freedom with affine transformation, which has only six independent parameters. The bilinear transform has eight free parameters. Hence, it is suitable for use with rectangular or quadrilateral patches. Generalized block matching is usually only adaptively used for those blocks where standard block matching is not satisfactory for avoiding imposed computational load.

16.4 RATE CONTROL

16.4.1 Introduction of Rate Control

The purpose of rate control is to optimize the perceived picture quality and to achieve a given constant average bit rate by controlling the allocation of the bits. From the viewpoint of rate control, the encoding can be classified into VBR coding and constant bit rate (CBR) coding. The VBR coding can provide a constant picture quality with variable coding bit rate, while the CBR will provide a constant bit rate with a nonuniform picture quality. Rate control and buffer regulation is an important issue for both VBR and CBR applications. In the case of VBR encoding, the rate controller attempts to achieve optimum quality for a given target rate. In the case of CBR encoding and real-time application, the rate control scheme has to satisfy the low-latency and VBV (video buffering verifier) buffer constraints. The VBV is a hypothetical decoder, which is conceptually connected to the output of an encoder (see Appendix C of ISO/IEC, 1995). The bitstream generated by the encoder is placed into the VBV buffer at the CBR rate that is being used. The rate control has to assure that the VBV will not be overflow or underflow. In addition, the rate control scheme has to be applicable to a wide variety of sequences and bit rates. At the GOP level, the total number of available bits is allocated among the various picture types, taking into account the constraints of the decoder buffer, so that the perceived quality is balanced. Within each picture, the available bits are allocated among the macroblocks to maximize the visual quality and to achieve the desired target of encoded bits for the whole picture.

16.4.2 Rate Control of Test Model 5 for MPEG-2

As we described before, the standard only defines the syntax for decoding. The test model is an example of the encoder, which may not be optimal; however, it can provide a compliant compressed bitstream. Also, the test model served as a reference during the development of the standard. The TM5 rate control algorithm consists of three steps to adapting the macroblock quantization parameter for controlling the bit rate.

16.4.2.1 Step 1: Target Bit Allocation

The target bit allocation is the first step of rate control. Before coding a picture, we need to estimate the number of bits available for coding this picture. The estimation is based on several factors. These include the picture type, buffer fullness, and picture complexity. The estimation of picture complexity is based on the number of bits and quantization parameter used for coding the same type of previous picture in the GOP. The initial complexity values are given according to the type of picture:

$$\begin{aligned} X_i &= 160 * \text{bit-rate}/115 \\ X_p &= 60 * \text{bit-rate}/115 \\ X_b &= 42 * \text{bit-rate}/115, \end{aligned} \tag{16.6}$$

where the subscripts i, p, and b stand for picture types I, P, and B (this will be applied to the formulas in this section). After a picture of a certain type (I, P, or B) is encoded, the respective "global complexity measure" (X_i, X_p, and X_b) is updated as

$$X_i = S_i Q_i, X_p = S_p Q_p, \text{ and } X_b = S_b Q_b, \tag{16.7}$$

where S_i, S_p, S_b are the number of bits generated by encoding this picture and Q_i, Q_p, Q_b are the average quantization parameters computed the actual quantization values used during the encoding of all the macroblocks including the skipped macroblocks. This estimation is very intuitive since, if the picture is more complicated, more bits are needed to encode it. The quantization parameter (step or interval) is used to normalize this measure because the number of bits generated by the encoder is inversely proportional to the quantization step. The quantization step can also be considered as a measure of coded picture quality. The target number of bits for the next picture in the GOP (T_i, T_p, and T_b) is computed as follows:

$$\begin{aligned} T_i &= \max\left\{ \frac{R}{1 + \frac{N_p X_p}{X_i K_p} + \frac{N_b X_b}{X_i K_b}}, \textit{bit-rate}/8 * \textit{picture-rate} \right\}, \\ T_p &= \max\left\{ \frac{R}{N_p + \frac{N_b K_p X_b}{X_b K_p}}, \textit{bit-rate}/8 * \textit{picture-rate} \right\} \\ T_b &= \max\left\{ \frac{R}{N_b + \frac{N_p K_p X_p}{X_p K_p}}, \textit{bit-rate}/8 * \textit{picture-rate} \right\}, \end{aligned} \tag{16.8}$$

where K_p and K_b are "universal" constants dependent on the quantization matrices. For the matrices of TM5, K_p = 1.0 and K_b = 1.4. The R is the remaining number of bits assigned to the GOP and after coding the picture this number is updated by subtracting the bit used for the picture. N_p and N_b are the number of P-pictures and B-pictures remaining in the current GOP in the encoding order. The problem of the above target bit assignment algorithm is that it does not handle scene changes efficiently.

16.4.2.2 Step 2: Rate Control

Within a picture, the bits used for each macroblock is determined by the rate control algorithm. Then a quantizer step is derived from the number of bits available for the macroblock to be coded. The following is an example of rate control for P-picture.

In Figure 16.20, d_0^p is initial virtual buffer fullness, the T_p is the target bits for P-picture. B_j is the number of bits generated by encoding all macroblocks in the picture up to and including jth macroblock. MB_cnt is the number of macroblocks in the picture. Before encoding the jth macroblock the virtual buffer fullness is adjusted during the encoding according to the following equation for the P-picture:

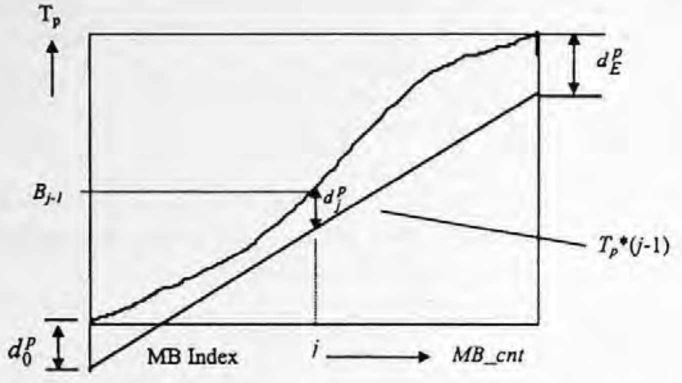


FIGURE 16.20 Rate control for P-picture. (From ISO/IEC, MPEG-2, Test Model 5, April 1993. With permission.)

$$d_j^p = d_0^p + B_{j-1} - \frac{T_p(j-1)}{MB\_cnt}. \tag{16.9}$$

Then the quantization step is computed with the equation:

$$Q_j^p = \frac{d_j^p}{r}, \tag{16.10}$$

where the "reaction parameter" r is given by $r = 2 * bit\text{-}rate/picture\text{-}rate$ and d_j^p is the fullness of the appropriate virtual buffer. This procedure is shown in Figure 16.20. The fullness of the virtual buffer for the last macroblock is used for encoding the next picture of the same type as the initial fullness.

The above example can be extended to the general case for all I-, P-, and B-pictures. Before encoding the jth macroblock, we compute the fullness of the appropriate virtual buffer:

$$\begin{aligned} d_j^i &= d_0^i + B_{j-1} - \frac{T_i(j-1)}{MB\_cnt} \quad \text{or} \\ d_j^p &= d_0^p + B_{j-1} - \frac{T_p(j-1)}{MB\_cnt} \quad \text{or} \\ d_j^b &= d_0^b + B_{j-1} - \frac{T_b(j-1)}{MB\_cnt}. \end{aligned} \tag{16.11}$$

Depending on the picture type, where d_0^i, d_0^p, d_0^b are initial fullness of the virtual buffers and d_j^i d_j^p, d_j^b are the fullness of virtual buffer at jth macroblock — one for each picture type. From the number of bits of the virtual buffer fullness, we compute the quantization step Q_j for macroblock j according to the buffer fullness:

$$Q_j = \frac{d_j * 31}{r}. \tag{16.12}$$

The initial values of the virtual buffer fullness are

$$\begin{aligned} d_0^i &= 10 \cdot r/31 \\ d_0^p &= K_p \cdot d_0^i \\ d_0^b &= K_b \cdot d_0^i \end{aligned} \tag{16.13}$$

K_p and K_b are constants which are defined in Equation 16.8.

16.4.2.3 Step 3: Adaptive quantization

Adaptive quantization is the last step of the TM5 rate control. It is noted that for active areas or busy areas, the human eyes are not so sensitive to the quantization noise, while the smooth areas are more sensitive to the quantization noise as discussed in Chapter 1. Based on this observation we modulate the quantization step obtained from the previous step in such a way to increase the quantization step for active areas and reduce the quantization step for the smooth areas. In other words, we use more bits in the smooth areas and fewer bits for the active areas. The experiment results have shown that the subjective quality is higher with the adaptive quantization step than without this step. The procedure of adaptive quantization in TM5 is as follows. First, the spatial activity measure for the *j*th macroblock is calculated from the four luminance frame–organized subblocks and the four luminance field–organized blocks using the intrapixel values:

$$act_j = 1 + \underset{sblk=1,8}{\mathrm{Min}}\left(var\_sblk\right), \tag{16.14}$$

where *var_sblk* is the variance of each spatial 8 × 8 block, which value is calculated as

$$var\_sblk = \frac{1}{64}\sum_{k=1}^{64}\left(P_k - P_{mean}\right)^2 \tag{16.15}$$

and P_k is the pixel value in the original 8 × 8 block and P_{mean} is the mean value of the block which is calculated as

$$P_{\mathrm{mean}} = \frac{1}{64}\sum_{k=1}^{64} P_k. \tag{16.16}$$

The normalized activity factor N_act_j is

$$N\_act_j = \frac{2 \cdot act_j + avg\_act}{act_j + 2 \cdot avg\_act} \tag{16.17}$$

where *avg_act* is the average value of act_j the last picture to be encoded. Therefore, this value will not give good results when a scene change occurs. On the first picture, this parameter takes the value of 400. Finally, we can obtain the modulated quantization step for *j*th macroblock:

$$mquant_j = Q_j \cdot N\_act_j \tag{16.18}$$

where Q_j is the reference quantization step value obtained in the last step. The final value of $mquant_j$ is clipped to the range of [1,31] and is used and coded as described in the MPEG standard.

As we indicated before, the TM5 rate control provides only a reference model. It is not optimized in many aspects. Therefore, there is still a lot of room for improving the rate control algorithm, such as to provide more precise estimation of average activity by preprocessing. In the following section, we will investigate the optimization problem for mode decision combined with rate control, which can provide a significant quality improvement as shown by experimental results.

16.5 OPTIMUM MODE DECISION

16.5.1 PROBLEM FORMATION

This section addresses the problem of determining the optimal MPEG (ISO/IEC, 1995) coding strategy in terms of the selection of macroblock coding modes and quantizer scales. In the TM5, the rate control operates independently from the coding mode selection for each macroblock. The coding mode is decided based only upon the energy of predictive residues. Actually, the two processes, coding mode decision and rate control, are intimately related to each other and should be determined jointly in order to achieve optimal coding performance. A constrained optimization problem can be formulated based on the rate–distortion characteristics, or $R(D)$ curves, for all the macroblocks that compose the picture being coded. Distortion for the entire picture is assumed to be decomposable and expressible as a function of individual macroblock distortions, with this being the objective function to minimize. The determination of the optimal solution is complicated by the MPEG differential encoding of motion vectors and dc coefficients, which introduce dependencies that carry over from macroblock to macroblock for a duration equal to the slice length. As an approximation, a near-optimum greedy algorithm can be developed. Once the upper bound in performance is calculated, it can be used to assess how well practical suboptimum methods perform.

Prior related work dealing with dependent quantization for MPEG include the work done by Ramchandran et al. (1994) and Lee and Dickerson (1994). Those works treated the problem of bit allocation where there is temporal dependency in coding complexity across I-, P-, and B-frames. While these techniques represent the most proper bit allocation strategies across frames from a theoretical viewpoint, no practical real-time MPEG encoding system will use even those proposed simplified techniques because they require an unwieldy number of preanalysis encoding passes over the window of dependent frames (one MPEG GOP). To overcome these computational burdens, more pragmatic solutions that can realistically be implemented have been considered by Sun et al. (1997). In this work, the major emphasis is not on the problem of bit allocation among I-, P-, and B-frames; rather, the authors choose to utilize the frame-level allocation method provided by the TM5. In this way, frame-level coding complexities are estimated from past frames without any forward preanalysis knowledge of future frames. This type of analysis forms the most reasonable set of assumptions for a practical real-time encoding system. Another method that extends the basic TM5 idea to alter frame budgets heuristically in the case of scene changes, use of dynamic GOP size, and temporal masking effects can be found in Wang (1995). These techniques also offer very effective and practical solutions for implementation. Given the chosen method for frame-level bit budget allocation, the focus of this section is to optimize macroblock coding modes and quantizers jointly within each frame.

There exists many choices for the macroblock coding mode under the MPEG-2 standard for P- and B-pictures, including intramode, no-motion-compensation mode, frame/field/dual-prime motion compensation intermode, forward/backward/average intermode, and field/frame DCT mode. In the standard TM5 reference (ISO/IEC, 1993), the coding mode for each macroblock is selected by comparing the energy of predictive residuals. For example, the intra/inter decision is determined by a comparison of the variance of the macroblock pixels against the variance of the predictive

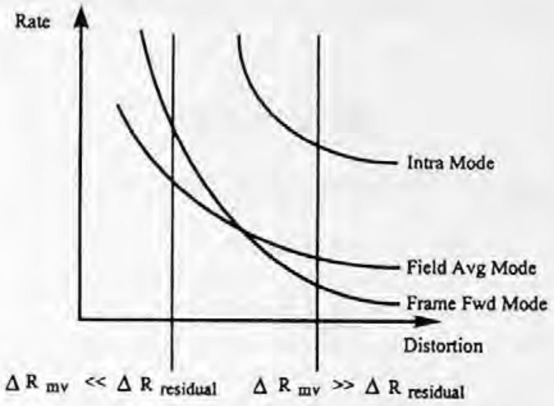


FIGURE 16.21 R(D) curves for different macroblock coding modes.

residuals; the interprediction mode is selected to be the intermode that has the least predictive residual MSE. The coding mode selected by the TM5 criteria does not result in the optimal coding performance.

In attempting to achieve optimal coding performance, it is important to realize that coding modes should be determined jointly with rate control because the best coding mode depends upon the operating point for rate. In deciding which of the various coding modes is best, one should consider what the operating point is for distortion, and also consider the trade-off between spending bits for coding the prediction residuals and bits for coding motion vectors.

The number of bits used for coding the macroblock is the sum of bits used for coding motion vectors and bits used for coding residuals:

$$R_{MB} = R_{mv} + R_{residual} \tag{16.19}$$

For example, in Figure 16.21, consider the decision between (1) frame-mode forward prediction and (2) field-mode bidirectional prediction. Mode (2) will almost always produce a prediction that has lower MSE than mode (1). However, mode (1) requires coding of fewer motion vectors than mode (2). Which mode is best? The answer depends on the operating point for distortion. When coding at a very coarse quant scale, mode (1) can perform better than mode (2) because the difference in bits required for coding motion vectors between the two modes may be much greater than the difference in bits required for coding residuals between the two modes. However, when coding at a fine quant scale, mode (2) can perform better than mode (1) because mode (2) provides a better prediction and the bits required for motion vectors would become negligible compared with bits for coding residuals.

Coding mode decisions and rate control can be determined jointly and optimally starting from the basics of constrained optimization using $R(D)$ curves. This optimal solution would be an *a posteriori* solution that assumes complete knowledge of $R(D)$. We investigate an optimal solution for objective functions of the form:

$$D_{PICT} = \sum_{i=1} D_{MBi}, \tag{16.20}$$

which states that the distortion for the picture, D_{PICT}, can be measured as an accumulation of individual macroblock distortions, D_{MB}, for all NMB number of macroblocks in the picture. We minimize this objective function subject to having individual macroblock distortions being uniform over the picture:

$$D_1 = D_2 = \cdots = D_{NMB} \tag{16.21}$$

and having the bits generated from coding each macroblock, R_{MB}, sum to a target bit allocation for the entire picture, R_{PICT}:

$$\sum_{i=1} R_{MBi} = R_{PICT} \tag{16.22}$$

The choice for the macroblock distortion measure, D_{MB}, can be the MSE computed over the pixels in the macroblock, or it can be a measure that reflects subjective distortion more accurately, such as luminance- and frequency-weighted MSE. Other choices for D_{MB} may be the quantizer scale used for coding the macroblock, or, better yet, the quantizer scale weighted by an activity-masking factor. In this chapter, we select distortion for each macroblock i to be a spatial-masking-activity-weighted quantizer scale:

$$D_{MBi} = qscale_i / N\_act_i, \tag{16.23}$$

where $N\_act_i \in [0.5, 2.0]$ is the normalized spatial masking activity quantizer weighting factor, as defined in the TM5:

$$N\_act_i = \frac{2 * act_i + avg\_act}{act_i + 2 * avg\_act}, \tag{16.24}$$

where act_i is the minimum luma block spatial variance for macroblock i and avg_act is the average value of act_i over the last picture to be coded. N_act_i reflects the relative amount of quantization error that can be tolerated for macroblock i as compared with the rest of the macroblocks that compose the picture. N_act_i depends strongly on whether the macroblock belongs to a smooth, edge, or textured region of the picture. Hence, the macroblock distortion metric is space variant and depends on the context of the local picture characteristics surrounding each macroblock. We assume that maintaining the same D_{MBi} for all macroblocks, or selecting the quantizer scales directly proportional to N_act_i in such a manner, corresponds to maintaining uniform subjective quality throughout the picture. The masking-activity-weighted quantizer scale is a somewhat coarse measure for image quality, but it reflects subjective image quality better than MSE or PSNR (peak signal-to-noise ratio), and it is a practical metric to compute that lends itself to an additive form for distortion.

It is important to note that the resulting distortion measure for the picture D_{PICT} is really only meaningful as a relative comparison figure for the same identical picture (thus having the same masking activities) quantized different ways. It is not useful comparing two different images. PSNR is only useful in this sense too, although with poorer subjective accuracy.

In the following, a procedure for obtaining the optimal coding performance with the joint optimization of coding mode selection and rate control is discussed. Since this method would be too complex to implement, a practical suboptimal heuristic algorithm is presented. Some simulation results and comparisons between the different algorithms — TM5 algorithm, near-optimum algorithm, and the practical suboptimum algorithm are also provided to assist the reader in understanding the differences in performance.

16.5.2 Procedure for Obtaining the Optimal Mode

16.5.2.1 Optimal Solution

The solution to the optimization problem is unique because the objective function is monotonic and the individual macroblock $R(D)$ functions are also monotonic. In order to solve for the optimal set of macroblock modes and quant-scales for the picture ($\overrightarrow{\text{mode}}$ and $\overrightarrow{\text{qscale}}$), the differential encoding of motion vectors and intra-dc coefficients as done in MPEG should be accounted for. According to MPEG, each slice has its own differential encoding chain. At the start of each slice, prediction motion vectors are reset to zero. As each macroblock is encoded in raster scan order, the macroblock motion vectors are encoded differentially with respect to prediction motion vectors that depend on the coding mode of the previous macroblock. These prediction motion vectors may be reset to zero in the case that the previous macroblock was coded as intra or skipped. Similarly, dc coefficients in continuous runs of intramacroblocks are encoded differentially with respect to the previous intramacroblock. The intra dc predictors are reset at the start of every slice, and at inter or skipped macroblocks. Slice boundaries delimit independent self-contained decodable units. Finding the optimal set of coding modes for the macroblocks in each slice entails a search through a trellis of dimensions S stages by M states per stage, with S being the slice size and M being the number of coding modes being considered (Figure 16.22). This trellis structure arises because there are M^2 distinct rate distortion, $R_{mode|previous\text{-}mode}(D)$, characteristic curves corresponding to each of M coding modes, with each in turn having a different dependency for each of M coding modes of the previous macroblock. We now consider populating the trellis links with values by sampling the set of these M^2S rate–distortion curves at a specific distortion level. For a given fixed macroblock distortion level, D_{MB}, each link on the trellis is assigned a cost equal to the number of bits to code a macroblock in a certain mode given the mode from which the preceding macroblock was coded. For any group of links entering a node, the cost of these links differs only because of the difference in bits caused by the motion vector and dc coefficient coding dependency upon the prior macroblock.

The computational requirements per slice involve:

- To determine link costs in the trellis, the number of "code the macroblock" operations (i.e., DCT + Quantization + RLC/VLC) is equal to M^2S.
- After determining all trellis link costs, the number of path searches is equal to M^S.

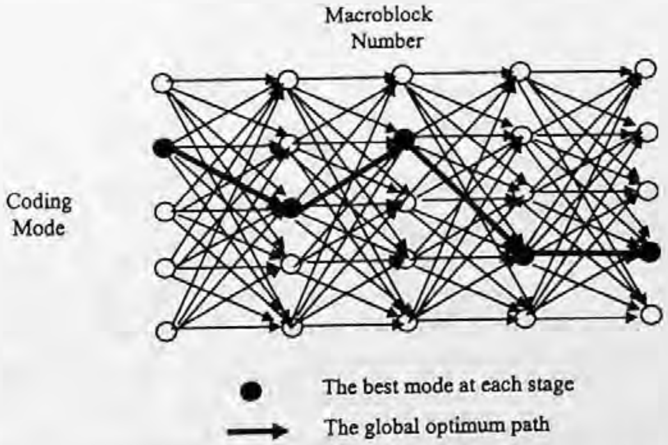


FIGURE 16.22 Full-search trellis, M^S (M is number of modes at each stage and S is the length of slice) searches needed to obtain the best path.

A general iterative procedure for obtaining the optimal solution is as follows:

1. Initialize a guess for $D_{MB} = D_{MB0}$. Since D_{MB} is the same for every macroblock in the picture, this sets an initial guess for the operating distortion level of the picture.
2. Perform for each slice in the picture:
   - For each macroblock in the slice and the mode considered, determine the quantizer scale which yields the distortion level D_{MB}, i.e., $q_s = f(D_{MB})$, where f is the function that describes the relationship between quantizer scale q_s and distortion D_{MB}. If we use the spatial-masking-activity-weighted quantizer scale as a measure of distortion (as from Equation 16.4), then q_s equals $N\_act * D_{MB}$.
   - Compute all the link costs in the trellis representing the slice. The link costs, R_{MBi} (mode k | mode j), represents the number of resulting bits (total bits for coding residual, motion vectors, and macroblock header) for coding macroblock i in mode k given that the preceding macroblock was coded in mode j.
   - Search through the trellis to find the path that has the lowest $\Sigma\, R_{MBi}$ over the slice.
3. Compute $\Sigma\, R_{MBi}$ for all macroblocks in the picture and compare to target R_{PICT}.
   - If $|\Sigma\, R_{MBi} - R_{PICT}| < \varepsilon$, then the optimal $\overrightarrow{\text{mode}}$ and $\overrightarrow{\text{qscale}}$ has been found for picture. Repeat the process for the next picture.
   - If $\Sigma\, R_{MBi} < R_{PICT}$, then decrement $D_{MB} = D_{MB} - \Delta D_{MB}$ and go to step 2.
   - If $\Sigma\, R_{MBi} > R_{PICT}$, then increment $D_{MB} = D_{MB} + \Delta D_{MB}$ and go to step 2.

16.5.2.2 Near-Optimal Greedy Solution

The solution from the full exponential-order search requires an unwieldy amount of computations. To avoid the heavy computational burden, we can use a greedy approach (Lee and Dickerson, 1994) to simplify and sidestep the dependency problems of the full-search method. In the greedy algorithm, the best coding mode selection for the current macroblock depends only upon the best mode of the previous coded macroblock. Therefore, the upper bound we obtain is a near-optimum solution instead of a global optimum. Figure 16.23 illustrates the greedy algorithm. After coding a macroblock in each of the M modes, the mode resulting in the least number of bits is chosen to be "best." The very next macroblock is coded with dependencies to that chosen "best" mode. The computations per slice are reduced to $M \times S$ "code the macroblock" operations and $M \times S$ comparisons. A general iterative procedure for obtaining the greedy solution is as follows:

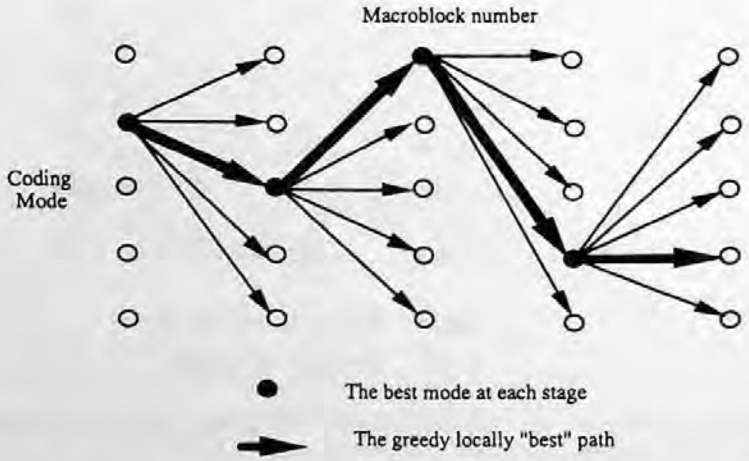


FIGURE 16.23 Greedy approach, $M \times S$ comparisons needed to obtain the locally "best" path.

1. Initialize a guess for $D_{MB} = D_{MBO}$.
2. Perform for each macroblock:
   - For each mode considered, determine the quantizer scale that yields the distortion level D_{MB}, i.e., $q_s = f(D_{MB})$, where f is the function we mentioned previously.
   - For each mode, code the macroblock in that mode with that q_s value and record the resulting number of generated bits, $R_{MBi(\text{mode } il \text{ mode } j)}$. The macroblock is coded based on the already determined mode of the preceding macroblock.
   - The "best" mode for macroblock i is the mode for which $R_{MBi(\text{mode } il \text{ mode } j) \text{ mode}}$ is smallest. This yields R_{MBi} bits for macroblock i.
3. Compute $\Sigma\, R_{MBi}$ for all macroblocks in the picture and compare to target R_{PICT}.
   - If $|\, \Sigma\, R_{MBi} - R_{PICT} \,| < \varepsilon$, then the optimal $\overrightarrow{\text{mode}}$ and $\overrightarrow{\text{qscale}}$ has been found for the picture. Repeat the process for the next picture.
   - If $\Sigma\, R_{MBi} < R_{PICT}$, then decrement $D_{MB} = D_{MB} - \Delta D_{MB}$ and go to step 2.
   - If $\Sigma\, R_{MBi} > R_{PICT}$, then increment $D_{MB} = D_{MB} + \Delta D_{MB}$ and go to step 2.

16.5.3 Practical Solution with New Criteria for the Selection of Coding Mode

It is obvious that the near-optimal solution discussed in the previous section is not a practical method because of its complexity. To determine the best mode, we have to know how many bits it takes to code each macroblock in every mode with the same distortion level. The total number of bits for each macroblock, R_{MB}, consists of three parts, bits for coding motion vectors, R_{mv}, bits for coding the predictive residue, R_{res}, and bits for coding macroblock header information, R_{header}, such as macroblock type, quantizer scale, and coded-block pattern.

$$R_{MB} = R_{mv} + R_{res} + R_{header}. \tag{16.25}$$

The number of bits for motion vectors, R_{mv}, can be easily obtained by VLC table lookup. But to obtain the number of bits for coding the predictive residue, one has to go through the three step coding procedure: (1) DCT, (2) quantization, and (3) VLC as shown in Figure 16.24. At step 3, R_{res} is obtained with a lookup table according to the run length of zeros and the level of quantized coefficients, i.e., R_{res} depends on the pair of values of run and level:

$$R_{res} = f(run, level). \tag{16.26}$$

As stated above, to obtain the upper-bound coding performance, all three steps are needed for each coding mode, and then the coding mode resulting in the least number of bits is selected as the best mode.

To obtain a much less computationally intensive method, it is preferred to use a statistical model of DCT coefficient bit usage vs. variance of the prediction residual and quantizer step size. This will provide an approximation of the number of residual bits, R_{res}. For this purpose we assume that the run and level pair in Equation 16.26 is strongly dependent on values of the quantizer scale, q_s, and the variance of the residue, V_{res}, for each macroblock. Intuitively, we would expect the number of bits to encode a macroblock is proportional to the variance of the residual and inversely

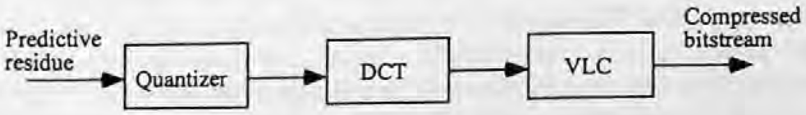


FIGURE 16.24 Coding stages to find bit count.

proportional to the value of quantizer step size. Therefore, a statistical model can be constructed by plotting R_{res} vs. the independent variables V_{res} and q_s over a large set of representative macroblock pixels from images typical of natural video material. This results in a scatter plot showing tight correlation, and hence a surface can be fit through the data points. It was found that Equation 16.26 can be approximately expressed as:

$$R_{res} \approx f\left(q_s, V_{res}\right) = \left(K/\left(Cq_s + q_s^2\right)\right) V_{res}, \tag{16.27}$$

where K and C are constants found through surface-fitting regression. If we assume R_{header} is a relatively fixed component that does not vary much with macroblock coding mode and can be ignored, then Equation 16.25 can be approximately replaced by:

$$R_{MB'} = R_{mv} + \left(K/\left(Cq_s + q_s^2\right)\right) V_{res}. \tag{16.28}$$

The value of $R_{MB'}$ reflects the variable portion of bit usage that is dependent on coding mode, and can be used as the measure for selecting the coding mode in our encoder. For a given quantizer step size, the mode resulting in the smallest value of $R_{MB'}$ is chosen as the "best" mode. It is obvious that, in the use of this new measurement to select the coding mode, the computational complexity increase over the TM5 is very slight (the same identical calculation for V_{res} is made in the TM5).

16.6 STATISTICAL MULTIPLEXING OPERATIONS ON MULTIPLE PROGRAM ENCODING

In this section, the strategies for statistical multiplexing operation on the multiple program encoding will be introduced. This topic is an extension of rate control into the case of multiple program encoding. First, a background survey of general encoding and multiplexing modes is reviewed. Second, the specific algorithm used in some current systems is introduced, its shortcomings are addressed, and possible amendments to the basic algorithm are described. Some potential research topics such as modeling strategies and methods for solving the problem are proposed for investigation. These topics may be good research topics for the interested graduate student.

16.6.1 Background of Statistical Multiplexing Operation

In many applications, several video sources may often be combined, or multiplexed, onto a single link for transmission. At the receiving end, the individual sources of data from the multiplexed data are demultiplexed and supplied to the intended receivers. For example, in an ATM network scenario many video sources originating from a local area are multiplexed onto a wide-area backbone trunk. In a satellite-broadcasting scenario, several video sources are multiplexed for transmission through a transponder. In a cable TV scenario, hundreds of video programs are broadcast onto a cable bus. Since the transmission channel, such as a trunk, a transponder, or a cable, is always an expensive resource, the limited channel capacity should be exploited as much as possible. The goal of statistical multiplexing encoding is to make the best use of the limited channel capacity possible. There are several approaches to encoding and multiplexing a plurality of video sources. In the following, we will compare the methods and describe the situation where each method is applicable. The qualitative comparisons are made in terms of trade-offs among factors of computation, implementation complexity, encoded picture quality, buffering delay, and channel utilization. To understand the statistical multiplexing method, we first introduce a simple case of deterministic multiplexing function of a CBR encoder. The standard method for performing the encoding and multiplexing function is to encode the source independently with a CBR. The

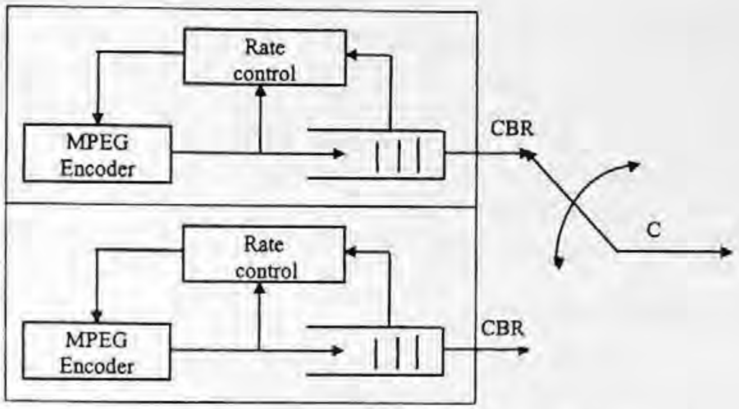


FIGURE 16.25 Independent encoding/muxing of CBR sources.

CBR encoder produces an encoded bit steam, representing the video supplied to it, at a predetermined CBR. To produce a CBR, the CBR encoder utilizes a rate buffer and feedback control mechanism that continually modifies the amount of quantization applied to the video signal, as shown in Figure 16.25.

The CBR encoder provides a CBR with varying encoded picture quality. This means that the degree of quantization applied depends upon the coding complexity of the current frame offered to the MPEG compression algorithm. Fine quantization is then applied to those frames that have low spatial and/or temporal coding complexity, and conversely coarse quantization is applied to frames that possess high spatial and temporal coding complexity in order to meet the bit rate. However, varying the quantization level corresponds to varying the video quality. Thus, in a CBR encoder, spatial and temporal complexity tends to be encoded in such a manner that the subjective quality of the reproduced image is lower than that of less complex images. This makes any form of rate control inherently bad in the sense that control is always imposed in a direction contrary to the goal of achieving uniform image quality. Usually, bit rates for CBR encoders are chosen so that the moderately difficult scenes can be coded to an acceptable quality level. Given that moderately difficult scenes give good results, then all simpler scenes will yield even better results with the given rate, while very difficult scenes will result in noticeable degradation. Since CBR encoders produce CBR, the multiplexing of a plurality of sources is very simple. The required channel capacity would simply be the sum of all the individual CBRs. Deterministic time or frequency division multiplexing of the individual CBR bitstreams onto the channel is a well-known and simple process. So with CBR encoding, uniformly consistent image quality is impossible for the video sequence with varying scene complexity, but the reward is the ease of multiplexing. The penalty of CBR coding with easy multiplexing may not only be nonuniform picture quality, also result in lower efficiency of channel bandwidth employment. Better efficiency can be gained by statistical multiplexing, whereby each source is encoded at a VBR coding approach. The VBR coding will result in uniform or consistent coded image quality by fixing the quantization scale or by modulating the quantization scale to a limited extent according to activity-masking attributes of the human visual system. Then, the bit rates generated by VBR coding vary with the coding complexity of the incoming video source material. Statistical multiplexing is referred to as StatMux. The coding gain of StatMux is possible through sharing of the channel resource jointly among the encoders. For example, two MPEG encoders may assign the appearance of their I-pictures at different time; this may reduce the limitation of the maximum channel bandwidth requirement since coding an I-picture may generate a large number of bits. This may not be a good example for practical applications. However, this explains that the process of StatMux is not a zero-sum game whereby one encoder's gain must be exactly another encoder's loss. In the process of StatMux, one encoder's gain is obtained by using the channel bandwidth that another encoder does not need at that time or that would bring a very marginal gain for another encoder at that time. More exactly, this concept of gains through sharing arises when the limited amount of bits is dynamically appropriated toward encoders that can best utilize those bits in substantially improving their image quality during complex segments and eschewed from encoders that can improve their image quality only marginally during easy segments. It is obvious that the CBR-encoded sources do not need statistical multiplexing since the bandwidth for each encoded source is well defined. The gain of statistical multiplexing is only possible with VBR-encoded sources. In the following section, we discuss two kinds of multiplexing with multiple VBR-encoded sources.

16.6.2 VBR Encoders in StatMux

There are two multiplexing methods for encoding multiple sources with VBR encoders, open loop and closed loop. Each VBR encoder in open-loop-multiplexing mode produces the most consistently uniform predefined image quality level regardless of the coding complexity of the incoming video sources. The image quality is decided by fixing the quantization scale. When the quantization scale is fixed, the SNR is fixed under assumption of white Gaussian quantization noise. Sometimes, the quantization scale is slightly modulated according to the image activity to match the human visual system, for example, in the method in MPEG-2 TM5. The resulting VBR bit rate process is generated by allowing the encoder to use freely however many bits needed to meet the predetermined quality level. Usually, each video source encoded by a VBR encoder in the open-loop mode is not geographically colocated and cannot be encoded jointly. However, the resulting VBR processes do share the channel "jointly," in the sense that the total channel bandwidth is not rigidly allocated among the sources in a fixed manner such as is done in CBR operation mode, where each source has the fixed portion of channel bandwidth. The instantaneous combined rates of all the VBR encoders may exceed the channel capacity, especially in the case when all the encoders generate bursts of bits at the same time. Then, the joint buffer will overflow, thereby leading to loss of data. However, there always still exists a possibility to utilize the channel capacity more efficiently by carefully allocating the loading conditions without loss of data. But totally open-loop VBR coding is not stationary and it is hard to achieve both good channel utilization and very limited data loss. A practical method of VBR transmission for use in the ATM environment involves placing limitations on the degree of variability allowed in VBR processes. Figure 16.26 illustrates the idea of self-regulating VBR encoders.

The difference between the proposed VBR encoder and a totally open-loop VBR encoder is that a looser form of rate control is imposed to the VBR encoder in order to avoid violating transmission constraints that are agreed to by the user and the network as part of the contract

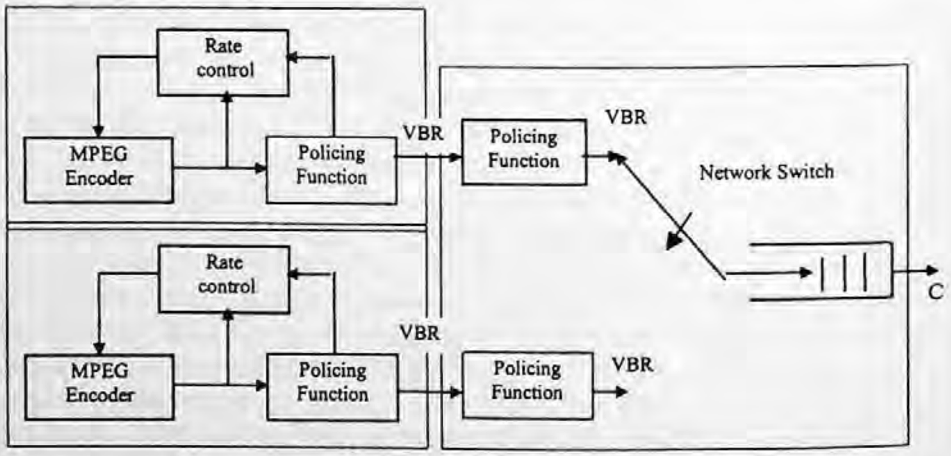


FIGURE 16.26 Independent encoding/muxing of geographically dispersed VBR sources.

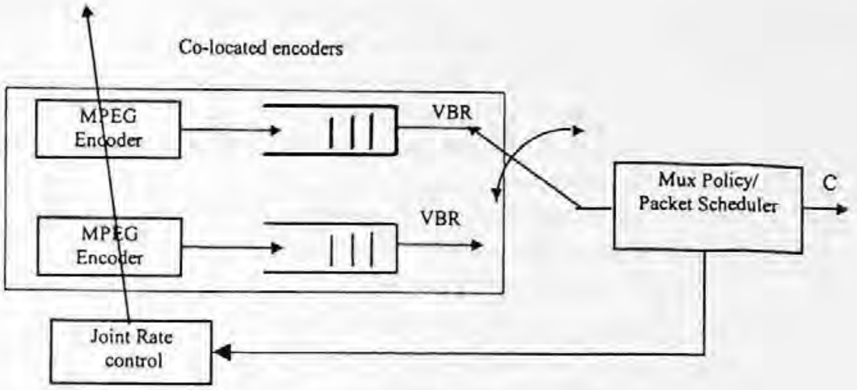


FIGURE 16.27 Method of joint rate control and multiplexing.

negotiated during the call setup stage. The rate control will match the policing function, which is enforced by the network. Looser rate control means that the rate control is not so strict as the one in the CBR case because it allows for the encoder to vary its output bit rate according to the coding complexity up to a certain degree as decided by the policing function.

In some applications such as the TV broadcasting or cable TV, the video sources may be geographically colocated at the same site. In such scenarios, additional gains can be realized by the StatMux in which the sources are jointly encoded and jointly multiplexed. By using a common rate controller, all encoders operate in VBR mode but without contending and stepping over one another as in independent VBR encoding and multiplexing. The joint rate controller assigns the total available channel capacity to each encoder so that a certain common quality level is maintained. The bit rates assigned to each individual encoder by joint rate control dynamically change based on the coding complexities of each video source to achieve the most uniform quality among the encoders and along the time for each encoder. In such a joint rate control method, although each encoder produces its own variable rate bits, the sum of bits produced by all encoders combined together is a CBR to fit the channel capacity. Such an idea is shown in Figure 16.27.

16.6.3 Research Topics of StatMux

The major problem of StatMux is how to allocate the bit rate resource among the video sources that share the common channel bit rate and are jointly encoded by a joint rate controller. This allocation should be based on the coding complexity of each source. The bit rate, $R_i(t)$, for encoder i at time t according to the normalized coding complexity of all encoders for the GOP period ending at time t, such as

$$R_i(t)=\frac{X_i(t)}{\sum_{j=1}^{N}X_j(t)}C, \tag{16.29}$$

where $X_i(t)$ is the coding complexity of source for encoder i at the time t over a GOP period and C is the total channel capacity. Also the bit rate assignment has to be updated from time to time to trace the variation of source complexity. In the following, we will discuss several topics which may be research topics for graduate students.

Forward Analysis — Without forward analysis, scene transitions are unanticipated and lead to incorrect bit allocation for a brief, transient period following the scene changes. If the bit allocation of a current video segment is based on the complexity of previous video segments and is adjusted by the available bit rate resource, those video segments which change from easy coding complexity to difficult coding complexity suffer the greatest degradation without preanalysis of upcoming increased complexity. Preanalysis could be performed with a dual set of encoders operating with a certain preprocessing delay ahead of the actual encoding process. As a simple example, we start to assign the equal portion of bit rate for each encoder; then we can obtain the average quantization scale for this GOP that can be considered as the forward analysis results of coding complexity. The real coding process can operate on the coding complexity obtained by the preanalysis. If we choose one or two GOPs according to the synchronous status of the input video sources to perform the preanalysis, it will result in small buffering delay.

Potential Modeling Strategies and Methods — Several modeling strategies and methods have been investigated to find a suitable procedure for classifying sources and determining what groups of sources can appropriately be jointly encoded together for transmission over a common channel to meet a specified image quality level. These modeling strategies and methods include modeling of video encoding, modeling of source coding complexity, and source classification. The modeling of a video-encoding algorithm involves measuring the operating performance of the individual encoders or characterizing its rate distortion function for a variety of scenes. Embodied into this model are the MPEG algorithms implemented for motion estimation, mode decision, rate control, and their joint optimization issues. It has been speculated that a hyperbolic functional form of

$$Rate = X/Distortion \tag{16.30}$$

would be appropriate over the normal operating bit rate range of 3 to 7 Mbps for MPEG-2 encoded CCIR601-sized videos. The hyperbolic shape of rate distortion curves would be also suitable for all video scenes. Actually, we can use a set of collected rate distortion data pairs with an encoder to fit a hyperbola through the points as shown in Figure 16.28 and estimate the shape parameter X. The value of X will be used to present the coding complexity offered to an encoder. For modeling at the GOP level, the rate would be the number of bits used to code that GOP and the distortion can be chosen as the averaging quantization scale over the GOP. In some of the literature, the distortion is taken as the average PSNR over the GOP or overall sequence. If it is assumed that the quantization noise is modeled by white Gaussian noise, then both distortion measures are equivalent.

After obtaining the correct coding complexity parameters, we can improve the StatMux algorithm by assigning an encoding bit budget to each encoder based on the GOP level normalized complexity measure X that each encoder is encoding. The GOP level normalized complexity measure $X(n)$ is defined as

$$X(n) = \sum_{i \in GOP} T(i)Q(i), \tag{16.31}$$

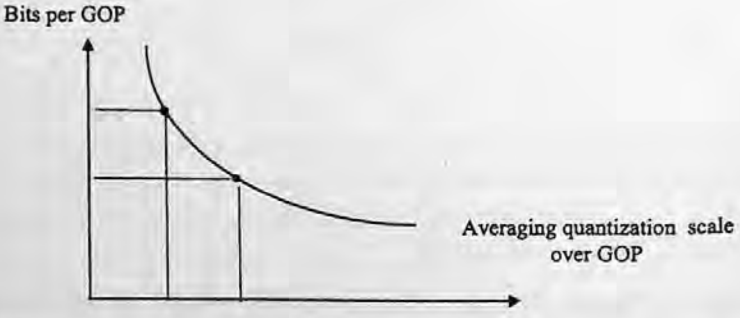


FIGURE 16.28 Rate distortion modeling of encoding algorithm and video source.

where n is the GOP number, $T(i)$ is the total number of bits used for encoding picture i, and $Q(i)$ is the average quantization scale used for encoding picture i. Some research results have shown that the $X(n)$ is insensitive to the operating bit rate, therefore, $X(n)$ is a reliable measure of the loading characteristics of a video source. Therefore, the study of accurate model of the random process of $x(n)$ is very important for improving the operations of the StatMux algorithm. The accurate model of $X(n)$ reflects the loading characteristics of the video source which dictates the share of total bit budget that an encoder expects to get. Several statistical models have been proposed to describe the complexity measure, $X(n)$. For example, an autoregressive process model is proposed for the intrascene $X(n)$ process. This proposed model is based on the following observations; the complexity measure within a single scene has a skewed distribution by the Gamma function, and furthermore, the complexity measure within a scene displays a strong temporal correlation and the form of the correlation is essentially exponential. The definition for the Mth order autoregressive model is

$$X(n)=\sum_{m=1}^{M} a(m)\bullet X(n-m)+e(n), \tag{16.32}$$

where $e(n)$ is the white noise process and $a(m)$ terms are the innovation filter coefficients. The statistics of the model such as the mean value, the variance, the correlation, and the marginal distribution are used to match those of actual signals by adjusting the $a(m)$ terms, $e(n)$ and M. Other cases, such as scene transition model, intercoded scene models, we leave as project topics for graduate students.

16.7 SUMMARY

In this chapter, the technical details of MPEG video are introduced. The technical detail of MPEG standards includes the decoding process of MPEG-1 and MPEG-2 video. Although the encoding process is not a standard part, it is very important for content and service providers. We discuss the most important parts of encoding techniques. Some examples such as the joint optimizing of mode decision and rate control are good examples to understand how the standard is used.

16.8 EXERCISES

16-1. According to your understanding, give several reasons to explain why the MPEG standards specify only decoding as a normative part and define encoding as an informative part (TM5).

16-2. Can an MPEG-2 video decoder decode a bitstream generated by an MPEG-1 video encoder? Summarize the main difference between the MPEG-1 and MPEG-2 video standards.

16-3. Prefiltering may reduce the noise of original video source and increase the coding efficiency. But at the same time prefiltering will result in a certain information loss. Conduct a project to investigate at what bit rate range prefiltering may benefit the coding efficiency for some video sources.

16-4. Use TM5 rate control to encode several video sequences (such as Flower Garden sequence) in two ways: (a) with adaptive quantization step, (b) without adaptive quantization step (Equation 6.16). Compare and discuss the numerical results and subjective results (observe the smooth areas carefully).

16-5. Why does MPEG-2 use different quantizer matrices for intra- and intercoding? Conduct a project to use different quantization matrices to encode several video sequences and report the results.

16-6. Conduct a project to encode several video sequences (a) with B-picture; (b) without B-picture. Compare the numerical and subjective results. Observe what difference exists between the sequences with fast motion and the sequence with slow motion. (Typical bit rates for CCIR601 sequences are 4 to 6 Mbps).

REFERENCES

Cano, D. and M. Benard, 3-D Kalman filtering of image sequences, in *Image Sequence Processing and Dynamic Scene Analysis,* T. S. Huang, Ed., Springer, Berlin, 1983, pp. 563-579.

Haskell, B. G., A. Puri, and A. N. Netravali, *Digital Video: Introduction to MPEG-2,* Chapman Hall, New York, 1997.

Isnardi, M. A. Consumers seek easy to use products, *IEEE Spectrum,* 64, 1993.

ISO/IEC 11172, International Standard, 1992.

ISO/IEC, MPEG-2 Test Model 5, ISO-IEC/JTC1/SC29/WG11, April, 1993.

ISO/IEC 13818 MPEG-2 International Standard, Video Recommendation ITU-T H.262, Jan. 10, 1995.

Katsaggelos, A. K., R. P. Kleihorst, S. N. Efstratiadis, and R. L. Lagendijk, Adaptive image sequence noise filtering methods, *Proceeding of SPIE Visual Communication and Image Processing,* Boston, Nov. 10-13, 1991.

Katto, J., J. Ohki, S. Nogaki, and M. Ohta, A wavelet codec with overlapped motion compensation for very low bit rate enviroment, *IEEE Trans. Circuits Syst. Video Technol.,* 4(3), 328-338, 1994.

Koc, U.-V. and K. J. R. Liu, DCT-based motion estimation, *IEEE Trans. Image Process.,* 7, 948-965, 1998.

Lee, J. and B.W. Dickerson, Temporally adaptive motion interpolation exploiting temporal masking in visual perception, *IEEE Trans. on Image Proc.,* 3(5), 513-526, 1994.

Mitchell, J. L., W. B. Pennebaker, C. E. Fogg, and D. J. LeGall, *MPEG Video Compression Standard,* Chapman Hall, New York, 1997.

Ozkan, M. K., M. I. Sezan, and A. M. Tekalp, Adaptive motion-compensated filtering of noisy image sequences, *IEEE Trans on Circuits and Systems for Video Technology,* 3(4), 277-290, 1993.

Ramchandran, K., A. Ortega, and M. Vetterli, Bit allocation for dependent quantization with application to MPEG video coders, *IEEE Trans. on Image Proc.,* 3(5), 533-545, 1994.

Sezan, M. I., M. K. Ozkan, and S. V. Fogel, Temporal adaptive filtering of noisy image sequences using a robust motion estimation algorithm, *IEEE ICASSP,* 2429-2432, 1991.

Sun, H. Sarnoff Internal technical report, May 1994.

Sun, H., W. Kwok, M. Chien, and C. H. John Ju, MPEG coding performance improvement by jointly optimization coding mode decision and rate control, *IEEE Trans. Circuits Syst. Video Technol.,* 7(3), 449-458, 1997.

Wang, L. Rate control for MPEG-2 video coding, *SPIE on Visual Communications and Image Processing,* pp. 53-64, Taipei, Taiwan, May 1995.

Young, R. W. and N. G. Kingsbury, Frequency-domain motion estimation using a complex lapped transform, *IEEE Trans. Image Process.,* 2(1), 2-17, 1993.

17 Application Issues of MPEG-1/2 Video Coding

This chapter is an extension of the previous chapter. We introduce several important application issues of MPEG-1/2 video which include the ATSC (Advanced Television Standard Committee) DTV standard which has been adopted by the FCC (Federal Communications Commission) as the TV standard in the United States, transcoding, down-conversion decoder, and error concealment.

17.1 INTRODUCTION

Digital video signal processing is an area of science and engineering that has developed rapidly over the past decade. The maturity of the moving picture expert group (MPEG) video-coding standard is a very important achievement for the video industry and provides strong support for digital transmission and storage of video signals. The MPEG coding standard is now being deployed for a variety of applications, which include high-definition television (HDTV), teleconferencing, direct broadcasting by satellite (DBS), interactive multimedia terminals, and digital video disk (DVD). The common feature of these applications is that the different source information such as video, audio, and data are all converted to the digital format and then mixed together to a new format which is referred to as the bitstream. This new format of information is a revolutionary change in the multimedia industry, since the digitized information format, i.e., the bitstream, can be decoded not only by traditional consumer electronic products such as television but also by the digital computer. In this chapter, we will present several application examples of MPEG-1/2 video standards, which include the ATSC DTV standard, transcoding, down-conversion decoder, and error concealment. The DTV standard is the application extension of the MPEG video standard. The transcoding and down-conversion decoders are the practical application issues which increase the features of compression-related products. The error concealment algorithms provide the tool for transmitting the compressed bitstream over noisy channels.

17.2 ATSC DTV STANDARDS

17.2.1 A Brief History

The birth of digital television (DTV) in the U.S. has undergone several stages: the initial stage, the competition stage, the collaboration stage, and the approval stage (Reitmeier, 1996). The concept of high-definition television (HDTV) was proposed in Japan in the late 1970s and early 1980s. During that period, Japan and Europe continued to make efforts in the development of analog television transmission systems, such as MUSE and HD-MAC systems. In early 1987, U.S. broadcasters fell behind in this field and felt they should take action to catch up with the new HDTV technology and petitioned the FCC to reserve a spectrum for terrestrial broadcasting of HDTV. As a result, the Advisory Committee on Advanced Television Service (ACATS) was founded in August 1987. This committee takes the role of recommending a standard to the FCC for approval. Thus, the process of selecting an appropriate HDTV system for the U.S. started. At the initial stage between 1987 and 1990, there were over 23 different analog systems proposed; among these systems two typical approaches were extended definition television (EDTV) which fits into a single 6-MHz channel and the high definition television (HDTV) approach which requires two 6-MHz channels. By 1990, ACATS had established the Advanced Television Test Center (ATTC), an official testing laboratory sponsored by broadcasters to conduct extensive laboratory tests in Virginia and field tests in Charlotte, NC. Also, the industry had formed the Advanced Television Standards Committee (ATSC) to perform the task of drafting the official standard documents of the selected winning system.

As we know, the current ATSC-proposed television standard is a digital system. In early 1990, the FCC issued a very difficult request to industry about the DTV standard. The FCC required the industry to provide full-quality HDTV service in a single 6-MHz channel. Having recognized the technical difficulty of this requirement at that time, the FCC also stated that this service could be provided by a simulcast service in which programs would be simultaneously broadcasted in both NTSC and the new television system. However, the FCC decided not to assign new spectrum bands for television. This means that simulcasting would occur in the already crowded VHF and UHF spectrum. The new television system had to use low-power transmission to avoid excessive interference into the existing NTSC services. Also, the new television system had to use a very aggressive compression approach to squeeze a full HDTV signal into the 6-MHz spectrum. One good thing was that backward compatibility with NTSC was not required. Actually, under these constraints the backward compatibility had already become impossible. Also, this goal could not be achieved by any of the previously proposed systems and it caused most of the competing proponents to reconsider their approaches. Engineers realized that it was almost impossible to use the traditional analog approaches to reach this goal and that the solution may be in digital approaches. After a few months of consideration, General Instrument announced its first digital system proposal for HDTV, DigiCigher, in June 1990. In the following half year, three other digital systems were proposed: the Advanced Digital HDTV by the Advanced Television Research Consortium, which included Thomson, Philips, Sarnoff, and NBC in November 1990; Digital Spectrum Compatible HDTV by Zenith and AT&T in December 1990; and Channel Compatible Digicipher by General Instrument and the Massachusetts Institute of Technology in January 1991. Thus, the competition stage started. The prototypes of four competing digital systems and the analog system, Narrow MUSE, proposed by NHK (Nippon Houson Kyokai, the Japan Broadcasting Corporation), were officially tested and extensively analyzed during 1992. After a first round of tests, it was concluded that the digital systems would be continued for further improvement and would be adopted. In February 1992, the ACATS recommended digital HDTV for the U.S. standard. It also recommended that the competing systems be either further improved and retested, or be combined into a new system. In the middle of 1993, the former competitors joined in a Grand Alliance. Then the DTV development entered the collaboration stage. The Grand Alliance began a collaborative effort to create the best system which combines the best features and capabilities of the formerly competing systems into a single "best of the best" system. After 1 year of joint effort by the seven Grand Alliance members, the Grand Alliance provided a new system that was prototyped and extensively tested in the laboratory and field. The test results showed that the system is indeed the best of the best compared with the formerly competing systems (Grand Alliance, 1994). The ATSC then recommended this system to the FCC as the candidate HDTV standard in the United States. During the following period, the computer industry realized that DTV provides the signals that can now be used for computer applications and the TV industry was invading its terrain. It presented different opinions about the signal format and was especially opposed to the interlaced format. This reaction delayed the approval of the ATSC standard. After a long debate, the FCC finally approved the ATSC standard in early 1997. But, the FCC did not specify the picture formats and leaves this issue to be decided by the market.

17.2.2 Technical Overview of ATSC Systems

The ATSC DTV system has been designed to satisfy the FCC requirements. The basic requirement is that no additional frequency spectrum will be assigned for DTV broadcasting. In other words, during a transition period, both NTSC and DTV service will be simultaneously broadcast on different channels and DTV can only use the taboo channels. This approach allows a smooth transition to DTV, such that the services of the existing NTSC receivers will remain and gradually be phased out of existence in the year 2006. The simulcasting requirement causes some technical difficulties in DTV design. First, the high-quality HDTV program must be delivered in a 6-MHz channel to make efficient use of spectrum and to fit allocation plans for the spectrum assigned to television broadcasting. Second, a low-power and low-interference signal must be used so that simulcasting in the same frequency allocations as current NTSC service does not cause excessive interference with the existing NTSC receiving, since the taboo channels are generally unsuitable for broadcasting an NTSC signal due to high interference. In addition to satisfying the frequency spectrum requirement, the DTV standard has several important features, which allow DTV to achieve interoperability with computers and data communications. The first feature is the adoption of a layered digital system architecture. Each individual layer of the system is designed to be interoperable with other systems at the corresponding layers. For example, the square pixel and progressive scan picture format should be provided to allow computers access to the compression layer or picture layer depending on the capacity of the computers and the ATM-like packet format for the ATM network to access the transport layer. Second, the DTV standard uses a header/descriptor approach to provide maximum flexible operating characteristics. Therefore, the layered architecture is the most important feature of DTV standards. The additional advantage of layering is that the elements of the system can be combined with other technologies to create new applications. The system of DTV standard includes four layers: the picture layer, the compression layer, the transport layer, and the transmission layer.

17.2.2.1 Picture Layer

At the picture layer, the input video formats have been defined. The Executive Committee of the ATSC has approved release of statement regarding the identification of the HDTV and Standard Definition Television (SDTV) transmission formats within the ATSC DTV standards. There are six video formats in the ATSC DTV standard, which are "High Definition Television." These formats are listed in Table 17.1.

The remaining 12 video formats are not HDTV format. These formats represent some improvements over analog NTSC and are referred to as "SDTV." These are listed in Table 17.2.

These definitions are fully supported by the technical specifications for the various formats as measured against the internationally accepted definition of HDTV established in 1989 by the ITU and the definitions cited by the FCC during the DTV standard development process. These formats cover a wide variety of applications, which include motion picture film, currently available HDTV production equipment, the NTSC television standard, and computers such as personal computers and workstations. However, there is no simple technique which can convert images from one pixel

TABLE 17.1
HDTV Formats

Spatial Format (X × Y active pixels)	Aspect Ratio	Temporal Rate (Hz progressive scan)
1920 × 1080 (square pixel)	16:9	23.976/24
		29.97/30
		59.94/60
1280 × 720 (square pixel)	16:9	23.976/24
		29.97/30
		59.94/60

TABLE 17.2
SDTV Formats

Spatial Format (X × Y active pixels)	Aspect Ratio	Temporal Rate (Hz progressive scan)
704 × 480 (CCIR601)	16:9 or 4:3	23.976/24 29.97/30 59.94/60
640 × 480 (VGA, square pixel)	4:3	23.976/24 29.97/30 59.94/60

format and frame rate to another that achieve interoperability among film and the various worldwide television standards. For example, all low-cost computers use square pixels and progressive scanning, while current television uses rectangular pixels and interlaced scanning. The video industry has paid a lot of attention to developing format-converting techniques. Some techniques such as deinterlacing, down/up-conversion for format conversion have already been developed. It should be noted that the broadcasters, content providers, and service providers can use any one of these DTV format. This results in a difficult problem for DTV receiver manufacturers who have to provide all kinds of DTV receivers to decode all these formats and then to convert the decoded signal to its particular display format. On the other hand, this requirement also gives receiver manufacturers the flexibility to produce a wide variety of products that have different functionality and cost, and the consumers freedom to choose among them.

17.2.2.2 Compression Layer

The raw data rate of HDTV of 1920 × 1080 × 30 × 16 (16 bits per pixel corresponds to 4:2:2 color format) is about 1 Gbps. The function of the compression layer is to compress the raw data from about 1 Gbps to the data rate of approximately 19 Mbps to satisfy the 6-MHz spectrum requirement. This goal is achieved by using the main profile and high level of the MPEG-2 video standard. Actually, during the development of the Grand Alliance HDTV system, many research results were adopted by the MPEG-2 standard at the same time; for example, the support for interlaced video format and the syntax for data partitioning and scalability. The ATSC DTV standard is the first and most important application example of the MPEG-2 standard. The use of MPEG-2 video compression fundamentally enables ATSC DTV devices to interoperate with MPEG-1/2 computer multimedia applications directly at the compressed bitstream level.

17.2.2.3 Transport Layer

The transport layer is another important issue for interoperability. The ATSC DTV transport layer uses the MPEG-2 system transport stream syntax. It is a fully compatible subset of the MPEG-2 transport protocol. The basic function of the transport layer is to define the basic format of data packets. The purposes of packetization include:

- Packaging the data into the fixed-size cells or packets for forward error correction (FEC) encoding to protect the bit error due to the communication channel noise;
- Multiplexing the video, audio, and data of a program into a bitstream;
- Providing time synchronization for different media elements;
- Providing flexibility and extensibility with backward compatibility.

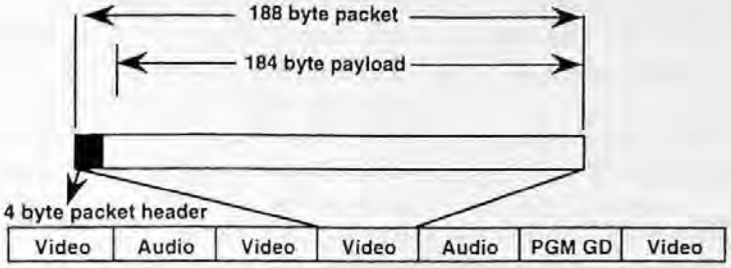


FIGURE 17.1 Packet structure of ATSC DTV transport layer.

The transport layer of ATSC DTV uses a fixed-length packet. The packet size is 188 bytes consisting of 184 bytes of payload and 4 bytes of header. Within the packet header, the13-bit packet identifier (PID) is used to provide the important capacity to combine the video, audio, and ancillary data stream into a single bitstream as shown in Figure 17.1. Each packet contains only a single type of data (video, audio, data, program guide, etc.) identified by the PID.

This type of packet structure packetizes the video, audio, and auxiliary data separately. It also provides the basic multiplexing function that produces a bitstream including video, five-channel surround-sound audio, and an auxiliary data capacity. This kind of transport layer approach also provides complete flexibility to allocate channel capacity to achieve any mix among video, audio, and other data services. It should be noted that the selection of 188-packet length is a trade-off between reducing the overhead due to the transport header and increasing the efficiency of error correction. Also, one ATSC DTV packet can be completely encapsulated with its header within four ATM packets by using 1 AAL byte per ATM header leaving 47 usable payload bytes times 4, for 188 bytes. The details of the transport layer is discussed in the chapter on MPEG systems.

Transmission Layer — The function of the transmission layer is to modulate the transport bitstream into a signal that can be transmitted over the 6-MHz analog channel. The ATSC DTV system uses a trellis-coded eight-level vestigial sideband (8-VSB) modulation technique to deliver approximately 19.3 Mbps in the 6-MHz terrestrial simulcast channel. VSB modulation inherently requires only processing the in-phase signal sampled at the symbol rate, thus reducing the complexity of the receiver, and ultimately the cost of implementation. The VSB signal is organized in a data frame that provides a training signal to facilitate channel equalization for removing multipath distortion. However, from several field-test results, the multipath distortion is still a serious problem of terrestrial simulcast receiving. The frame is organized into segments each with 832 symbols. Each transmitted segment consists of one synchronization byte (four symbols), 187 data bytes, and 20 R-S parity bytes. This corresponds to a 188-byte packet, which is protected by 20-byte R-S code. Interoperability at the transmission layer is required by different transmission media applications. The different media use different modulation techniques now, such as QAM for cable and QPSK for satellite. Even for terrestrial transmission, European DVB systems use OFDM transmission. The ATV receivers will not only be designed to receive terrestrial broadcasts, but also the programs from cable, satellite, and other media.

17.3 TRANSCODING WITH BITSTREAM SCALING

17.3.1 BACKGROUND

As indicated in the previous chapters, digital video signals exist everywhere in the format of compressed bitstreams. The compressed bitstreams of video signals are used for transmission and storage through different media such as terrestrial TV, satellite, cable, the ATM network, and the

Internet. The decoding of a bitstream can be implemented in either hardware or software. However, for high-bit-rate compressed video bitstreams, specially designed hardware is still the major decoding approach due to the speed limitation of current computer processors. The compressed bitstream as a new format of video signal is a revolutionary change to video industry since it enables many applications. On the other hand, there is a problem of bitstream conversion. Bitstream conversion or transcoding can be classified as bit rate conversion, resolution conversion, and syntax conversion. Bit rate conversion includes bit rate scaling and the conversion between constant bit rate (CBR) and variable bit rate (VBR). Resolution conversion includes spatial resolution conversion and temporal resolution conversion. Syntax conversion is needed between different compression standards such as JPEG, MPEG-1, MPEG-2, H.261, and H.263. In this section, we will focus on the topic of bit rate conversion, especially on bit rate scaling since it finds wide application and readers can extend the idea to other kinds of transcoding. Also, we limit ourselves to focus on the problem of scaling an MPEG CBR-encoded bitstream down to a lower CBR. The other kind of transcoding, down-conversion decoder, will be presented in a separate section.

The basic function of bitstream scaling may be thought of as a black box, which passively accepts a precoded MPEG bitstream at the input and produces a scaled bitstream, which meets new constraints that are not known *a priori* during the creation of the original precoded bitstream. The bitstream scaler is a transcoder, or filter, that provides a match between an MPEG source bitstream and the receiving load. The receiving load consists of the transmission channel, the destination decoder, and perhaps a destination storage device. The constraint on the new bitstream may be bound by a variety of conditions. Among them are the peak or average bit rate imposed by the communications channel, the total number of bits imposed by the storage device, and/or the variation of bit usage across pictures due to the amount of buffering available at the receiving decoder.

While the idea of bitstream scaling has many concepts similar to those provided by the various MPEG-2 scalability profiles, the intended applications and goals differ. The MPEG-2 scalability methods (data partitioning, SNR scalability, spatial scalability, and temporal scalability) are aimed at providing encoding of source video into multiple service grades (that are predefined at the time of encoding) and multitiered transmission for increased signal robustness. The multiple bitstreams created by MPEG-2 scalability are hierarchically dependent in such a way that by decoding an increasing number of bitstreams, higher service grades are reconstructed. Bitstream scaling methods, in contrast, are primarily decoder/transcoder techniques for converting an existing precoded bitstream to another one that meets new rate constraints. Several applications that motivate bitstream scaling include the following:

1. Video-On-Demand — Consider a video-on-demand (VOD) scenario wherein a video file server includes a storage device containing a library of precoded MPEG bitstreams. These bitstreams in the library are originally coded at high quality (e.g., studio quality). A number of clients may request retrieval of these video programs at one particular time. The number of users and the quality of video delivered to the users are constrained by the outgoing channel capacity. This outgoing channel, which may be a cable bus or an ATM trunk, for example, must be shared among the users who are admitted to the service. Different users may require different levels of video quality, and the quality of a respective program will be based on the fraction of the total channel capacity allocated to each user. To accommodate a plurality of users simultaneously, the video file server must scale the stored precoded bitstreams to a reduced rate before it is delivered over the channel to respective users. The quality of the resulting scaled bitstream should not be significantly degraded compared with the quality of a hypothetical bitstream so obtained by coding the original source material at the reduced rate. Complexity cost is not such a critical factor because only the file server has to be equipped with the bitstream scaling hardware, not every user. Presumably, video service providers would be willing to pay a high cost for delivering the possible highest-quality video at a prescribed bit rate.

As an option, a sophisticated video file server may also perform scaling of multiple original precoded bitstreams jointly and statistically multiplex the resulting scaled VBR bitstreams into the channel. By scaling the group of bitstreams jointly, statistical gains can be achieved. These statistical gains can be realized in the form of higher and more uniform picture quality for the same channel capacity. Statistical multiplexing over a DirecTv transponder (Isnardi, 1993) is one example of an application of video statistical multiplexing.

2. Trick-play Track on Digital VTRs — In this application, the video bitstream is scaled to create a sidetrack on video tape recorders (VTRs). This sidetrack contains very coarse quality video sufficient to facilitate trick-modes on the VTR (e.g., FF and REW at different speeds). Complexity cost for the bitstream scaling hardware is of significant concern in this application since the VTR is a mass consumer item subject to mass production.
3. Extended-Play Recording on Digital VTRs — In this application, video is broadcast to users' homes at a certain broadcast quality (~6 Mbps for standard-definition video and ~24 Mbps for high-definition video). With a bitstream scaling feature in their VTRs, users may record the video at a reduced rate, akin to extended-play (EP) mode on today's VHS recorders, thereby recording a greater duration of video programs onto a tape at lower quality. Again, hardware complexity costs would be a major factor here.

17.3.2 Basic Principles of Bitstream Scaling

As described previously, the idea of scaling an MPEG-2-compressed bitstream down to a lower bit rate is initiated by several applications. One problem is the criteria that should be used to judge the performance of an architecture that can reduce the size or rate of an MPEG-compressed bitstream. Two basic principles of bitstream scaling are (1) the information in the original bitstream should be exploited as much as possible, and (2) the resulting image quality of the new bitstream with a lower bit rate should be as close as possible to a bitstream created by coding the original source video at the reduced rate. Here, we assume that for a given rate the original source is encoded in an optimal way. Of course, the implementation of hardware complexity also has to be considered. Figure 17.2 shows a simplified encoding structure of MPEG encoding in which the rate control mechanism is not shown.

In this structure, a block of image data is first transformed to a set of coefficients; the coefficients are then quantized with a quantizer step which is decided by the given bit rate budget, or number of bits assigned to this block. Finally, the quantized coefficients are coded in variable-length coding to the binary format, which is called the bitstream or bits.

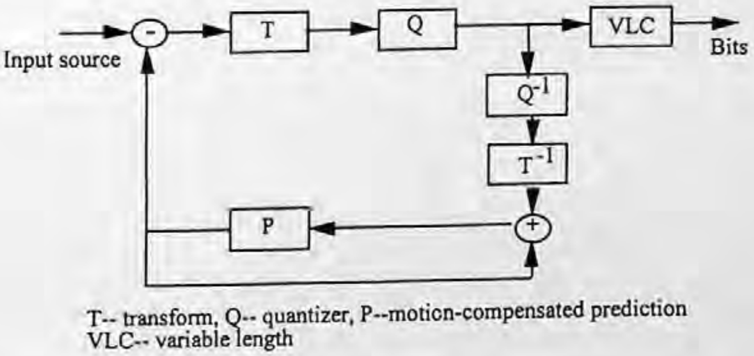

Input source
-
T
Q
VLC
Bits
Q-1
T-1
+
P
T-- transform, Q-- quantizer, P--motion-compensated prediction
VLC-- variable length

FIGURE 17.2 Simplified encoder structure. T = transform, Q = quantizer, P = motion-compensated prediction, VLC = variable length.

From this structure it is obvious that the performance of changing the quantizer step will be better than cutting higher frequencies when the same amount of rate needs to be reduced. In the original bitstream the coefficients are quantized with finer quantization steps which are optimized at the original high rate. After cutting the coefficients of higher frequencies, the rest of the coefficients are not quantized with an optimal quantizer. In the method of requantization all coefficients are requantized with an optimal quantizer which is determined by the reduced rate; the performance of the requantization method must be better than the method of cutting high frequencies to reach the reduced rate. The theoretical analysis is given in Section 17.3.4.

In the following, several different architectures that accomplish the bitstream scaling are discussed. The different methods have varying hardware implementation complexities; each has its own degree of trade-off between required hardware and resulting image quality.

17.3.3 Architectures of Bitstream Scaling

Four architectures for bitstream scaling are discussed. Each of the scaling architectures described has its own particular benefits that are suitable for a particular application.

Architecture 1: The bitstream is scaled by cutting high frequencies.
Architecture 2: The bitstream is scaled by requantization.
Architecture 3: The bitstream is scaled by reencoding the reconstructed pictures with motion vectors and coding decision modes extracted from the original high-quality bitstream.
Architecture 4: The bitstream is scaled by reencoding the reconstructed pictures with motion vectors extracted from the original high-quality bitstream, but new coding decisions are computed based on reconstructed pictures.

Architectures 1 and 2 are considered for VTR applications such as trick-play modes and EP recording. Architectures 3 and 4 are considered for and other applicable StatMux scenarios.

17.3.3.1 Architecture 1: Cutting AC Coefficients

A block diagram illustrating architecture 1 is shown in Figure 17.3a. The method of reducing the bit rate in architecture 1 is based on cutting the higher-frequency coefficients. The incoming precoded CBR stream enters a decoder rate buffer. Following the top branch leading from the rate buffer, a VLD is used to parse the bits for the next frame in the buffer to identify all the variable-length codewords that correspond to ac coefficients used in that frame. No bits are removed from the rate buffer. The codewords are not decoded, but just simply parsed by the VLD parser to determine codeword lengths. The bit allocation analyzer accumulates these ac bit counts for every macroblock in the frame and creates an ac bit usage profile as shown in Figure 17.3(b). That is, the analyzer generates a running sum of ac DCT coefficient bits on a macroblock basis:

$$PV_N = \sum AC\_BITS, \tag{17.1}$$

where PV_N is the profile value of a running sum of AC codeword bits until the macroblock N. In addition, the analyzer counts the sum of all coded bits for the frame, TB (total bits). After all macroblocks for the frame have been analyzed, a target value TV_{AC}, of ac DCT coefficient bits per frame is calculated as

$$TV_{AC} = PV_{LS} - \alpha * TB - B_{EX}, \tag{17.2}$$

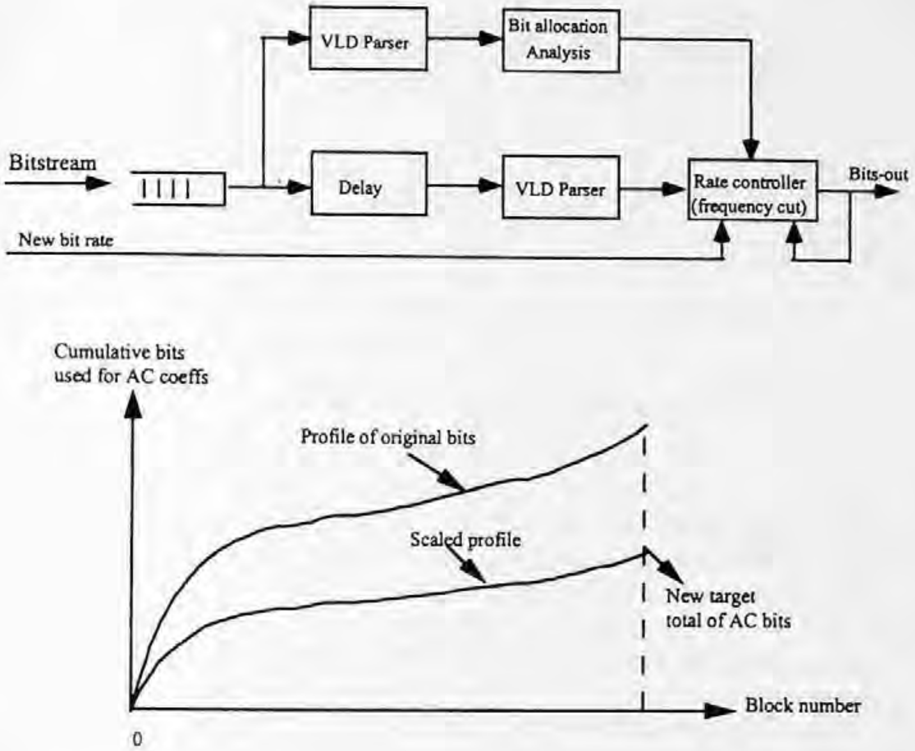


FIGURE 17.3 (a) Architecture 1: cutting high frequencies. (b) Profile map.

where TV_{AC} is the target value of AC codeword bits per frame, PV_{LS} is the profile value at the last macroblock, α is the percentage by which the preencoded bitstream is to be reduced, TB is the total bits, and B_{EX} is the amount of bits by which the previous frame missed its desired target. The profile value of AC coefficient bits is scaled by the factor TV_{AC}/PV_{LS}. Multiplying each PV_N performs scaling by that factor to generate the linearly scaled profile shown in Figure 17.3(b). Following the bottom branch from the rate buffer, a delay is inserted equal to the amount of time required for the top branch analysis processing to be completed for the current frame. A second VLD parser accesses and removes all codeword bits from the buffer and delivers them to a rate controller. The rate controller receives the scaled target bit usage profile for the amount of ac bits to be used within the frame. The rate controller has memory to store all coefficients associated with the current macroblock it is operating on. All original codeword bits at a higher level than ac coefficients (i.e., all fixed-length header codes, motion vector codes, macroblock type codes, etc.) are held in memory and will be remultiplexed with all AC codewords in that macroblock that have not been excised to form the outgoing scaled bitstream. The rate controller determines and flags in the macroblock codeword memory which AC codewords to keep and which to excise. AC codewords are accessed from the macroblock codeword memory in the order $AC11$, $AC12$, $AC13$, $AC14$, $AC15$, $AC16$, $AC21$, $AC22$, $AC23$, $AC24$, $AC25$, $AC26$, $AC31$, $AC32$, $AC33$, etc., where $ACij$ denotes the ith AC codewords from jth block in the macroblock if it is present. As the AC codewords are accessed from memory, the respective codeword bits are summed and continuously compared with the scaled profile value to the current macroblock, less the number of bits for insertion of EOB (end-of-block) codewords. Respective AC codewords are flagged as kept until the running sum of AC codewords bits exceeds the scaled profile value less EOB bits. When this condition is met, all remaining AC codewords are marked for being excised. This process continues until all macroblocks have their kept codewords reassembled to form the scaled bitstream.

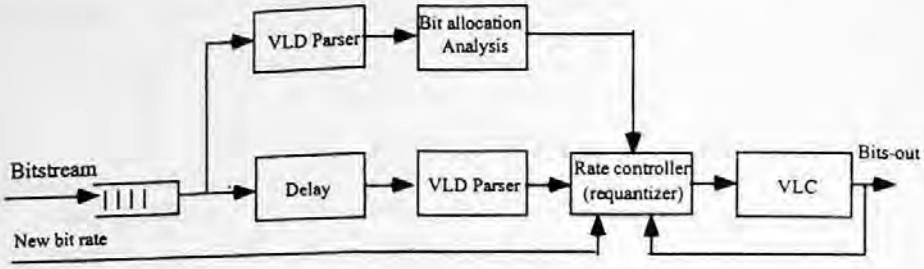


FIGURE 17.4 Architecture 2: increasing quantization step.

17.3.3.2 Architecture 2: Increasing Quantization Step

Architecture 2 is shown in Figure 17.4. The method of bitstream scaling in architecture 2 is based on increasing the quantization step. This method requires additional dequantizer/quantizer and variable-length coding (VLC) hardware over the first method. Like the first method, it also makes a first VLD pass on the bitstream and obtains a similar scaled profile of target cumulative codeword bits vs. macroblock count to be used for rate control.

The rate control mechanism differs from this point on. After the second-pass VLD is made on the bitstream, quantized DCT coefficients are dequantized. A block of finely quantized DCT coefficients is obtained as a result of this. This block of DCT coefficients is requantized with a coarser quantizer scale. The value used for the coarser quantizer scale is determined adaptively by making adjustments after every macroblock so that the scaled target profile is tracked as we progress through the macroblocks in the frame:

$$Q_N = Q_{NOM} + G * \left(\sum_{N-1} \left(BU - PV_{N-1} \right) \right), \tag{17.3}$$

where Q_N is the quantization factor for macroblock N, Q_{NOM} is an estimate of the new nominal quantization factor for the frame, $\Sigma_{N-1} BU$ is the cumulative amount of coded bits up to macroblock $N - 1$, and G is a gain factor which controls how tightly the profile curve is tracked through the picture. Q_{NOM} is initialized to an average guess value before the very first frame, and updated for the next frame by setting it to Q_{LS} (the quantization factor for the last macroblock) from the frame just completed. The coarsely requantized block of DCT coefficients is variable-length-coded to generate the scaled bitstream. The rate controller also has provisions for changing some macroblock-layer codewords, such as the macroblock-type and coded-block pattern to ensure a legitimate scaled bitstream that conforms to MPEG-2 syntax.

17.3.3.3 Architecture 3: Reencoding with Old Motion Vectors and Old Decisions

The third architecture for bitstream scaling is shown in Figure 17.5. In this architecture, the motion vectors and macroblock coding decision modes are first extracted from the original bitstream, and at the same time the reconstructed pictures are obtained from the normal decoding procedure. Then the scaled bitstream is obtained by reencoding the reconstructed pictures using the old motion vectors and macroblock decisions from the original bitstream. The benefits obtained from this architecture compared with full decoding and reencoding is that no motion estimation and decision computation is needed.

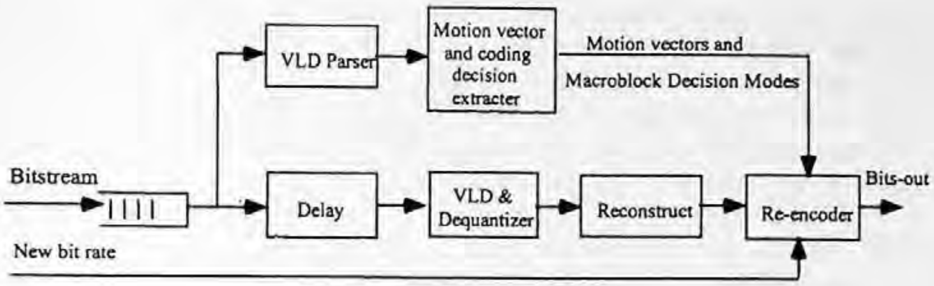


FIGURE 17.5 Architecture 3.

17.3.3.4 Architecture 4: Reencoding with Old Motion Vectors and New Decisions

Architecture 4 is a modified version of architecture 3 in which new macroblock decision modes are computed during reencoding based on reconstructed pictures. The scaled bitstream created this way is expected to yield an improvement in picture quality because the decision modes obtained from the high-quality original bitstream are not optimal for reencoding at the reduced rate. For example, at higher rates the optimal mode decision for a macroblock is more likely to favor bidirectional field motion compensation over forward frame motion compensation. But at lower rates, only the opposite decision may be true. In order for the reencoder to have the possibility of deciding on new macroblock coding modes, the entire pool of motion vectors of every type must be available. This can be supplied by augmenting the original high-quality bitstream with ancillary data containing the entire pool of motion vectors during the time it was originally encoded. It could be inserted into the user data every frame. For the same original bit rate, the quality of an original bitstream obtained this way is degraded compared with an original bitstream obtained from architecture 3 because the additional overhead required for the extra motion vectors steals away bits for actual encoding. However, the resulting scaled bitstream is expected to show quality improvement over the scaled bitstream from architecture 3 if the gains from computing new and more accurate decision modes can overcome the loss in original picture quality. Table 17.3 outlines the hardware complexity savings of each of the three proposed architectures as compared with full decoding and reencoding.

17.3.3.5 Comparison of Bitstream Scaling Methods

We have described four architectures for bitstream scaling which are useful for various applications as described in the introduction. Among the four architectures, architectures 1 and 2 do not require

TABLE 17.3
Hardware Complexity Savings over Full Decoding/Reencoding

Coding Method	Hardware Complexity Savings
Architecture 1	No decoding loop, no DCT/IDCT, no frame store memory, no encoding loop, no quantizer/dequantizer, no motion compensation, no VLC, simplified rate control
Architecture 2	No decoding loop, no DCT/IDCT, no frame store memory, no encoding loop, no motion compensation, simplified rate control
Architecture 3	No motion estimation, no macroblock coding decisions
Architecture 4	No motion estimation

entire decoding and encoding loops or frame store memory for reconstructed pictures, thereby saving significant hardware complexity. However, video quality tends to degrade through the group of pictures (GOP) until the next I-picture due to drift in the absence of decoder/encoder loops. For large scaling, say, for rate reduction greater than 25%, architecture 1 produces poor-quality blocky pictures, primarily because many bits were spent in the original high-quality bitstream on finely quantizing the dc and other very low-order ac coefficients. Architecture 2 is a particularly good choice for VTR applications since it is a good compromise between hardware complexity and reconstructed image quality. Architectures 3 and 4 are suitable for VOD server applications and other StatMux applications.

17.3.4 Analysis

In this analysis, we assume that the optimal quantizer is obtained by assigning the number of bits according to the variance or energy of the coefficients. It is slightly different from MPEG standard which will be explained later, but the principal concept is the same and the results will hold for the MPEG standard. We first analyze the errors caused by cutting high coefficients and increasing the quantizer step. The optimal bit assignment is given by Jayant and Noll (1984):

$$R_{k0} = R_{av0} + \frac{1}{2}\log_2 \frac{\sigma_k^2}{\left(\prod_{i=0}^{N-1}\sigma_i^2\right)^{1/N}}, \quad k = 0, 1, \ldots, N-1, \tag{17.4}$$

where N is the number of coefficients in the block, R_{k0} is the number of bits assigned to the kth coefficient, R_{av0} is the average number of bits assigned to each coefficient in the block, i.e., $R_{T0} = N \cdot R_{av0}$, is the total bits for this block under a certain bit rate, and σ_k^2 is the variance of kth coefficient. Under the optimal bit assignment (17.4), the minimized average quantizer error, σ_{q0}^2, is

$$\sigma_{q0}^2 = \frac{1}{N}\sum_{k=1}^{N-1}\sigma_{qk}^2 = \frac{1}{N}\sum_{k=1}^{N-1} 2^{-2R_{k0}} \cdot \sigma_k^2, \tag{17.5}$$

where σ_{qk}^2 is the quantizer error of kth coefficient. According to Equation 17.4, we have two major methods to reduce the bit rate, cutting high coefficients or decreasing the R_{av}, i.e., increasing the quantizer step. We are now analyzing the effects on the reconstructed errors caused by the method of cutting high-order coefficients. Assume that the number of the bits assigned to the block is reduced from R_{T0} to R_{T1}. Then the bits to be reduced, ΔR_1, are equal to $R_{T0} - R_{T1}$.

In the case of cutting high frequencies, say, the number of coefficients is reduced from N to M, then

$$R_{k0} = 0 \text{ for } K < M, \text{ and } \Delta R_1 = R_{T0} - R_{T1} = \sum_{k=M}^{N-1} R_{k0}. \tag{17.6}$$

the quantizer error increased due to the cutting is

$$\Delta\sigma_{q1}^2 = \sigma_{q1}^2 - \sigma_{q0}^2 = \frac{1}{N}\left(\sum_{k=0}^{M-1} 2^{-2R_{k0}} \cdot \sigma_k^2 + \sum_{k=M}^{N-1}\sigma_k^2 - \sum_{k=0}^{N-1} 2^{-2R_{k0}} \cdot \sigma_k^2\right) \tag{17.7}$$

$$= \frac{1}{N}\left(\sum_{k=M}^{N-1}\sigma_k^2 - \sum_{k=M}^{N-1} 2^{-2R_{k0}}\cdot\sigma_k^2\right)$$

$$= \frac{1}{N}\sum_{k=M}^{N-1}\left(1-2^{-2R_{k0}}\right)\cdot\sigma_k^2,$$

where σ_{q1}^2 is the quantizer error after cutting the high frequencies.

In the method of increasing quantizer step, or decreasing the average bits, from R_{av0} to R_{av2}, assigned to each coefficient, the number of bits reduced for the block is

$$\Delta R_2 = R_{T0} - R_{T2} = N\cdot\left(R_{av0} - R_{av2}\right) \tag{17.8}$$

and the bits assigned to each coefficient become now

$$R_{k2} = R_{av2} + \frac{1}{2}\log_2\frac{\sigma_k^2}{\left(\prod_{i=0}^{N-1}\sigma_i^2\right)^{1/N}}, \quad k = 0, 1, \ldots, N-1, \tag{17.9}$$

The corresponding quantizer error increased by the cutting bits is

$$\begin{aligned}\Delta\sigma_{q2}^2 = \sigma_{q2}^2 - \sigma_{q0}^2 &= \frac{1}{N}\left(\sum_{k=0}^{N-1}2^{-2R_{k2}}\cdot\sigma_k^2 - \sum_{k=0}^{N-1}2^{-2R_{k0}}\cdot\sigma_k^2\right)\\ &= \frac{1}{N}\sum_{k=0}^{N-1}\left(2^{-2R_{k2}} - 2^{-2R_{k0}}\right)\cdot\sigma_k^2,\end{aligned} \tag{17.10}$$

where σ_{q2}^2 is the quantizer error at the reduced bit rate.

If the same number of bits is reduced, i.e., $\Delta R_1 = \Delta R_2$, it is obvious that $\Delta\sigma_{q2}^2$ is smaller than $\Delta\sigma_{q1}^2$ since σ_{q2}^2 is the minimized value at the reduced rate. This implies that the performance of changing the quantizer step will be better than cutting higher frequencies when the same amount of rate needs to be reduced. It should be noted that in the MPEG video coding, more sophisticated bit assignment algorithms are used. First, different quantizer matrices are used to improve the visual perceptual performance. Second, different VLC tables are used to code the DC values and the AC transform coefficients and the run-length coding is used to code the pairs of the zero-run length and the values of amplitudes. However, in general, the bits are still assigned according to the statistical model that indicates the energy distribution of the transform coefficients. Therefore, the above theoretical analysis will hold for the MPEG video coding.

17.4 DOWN-CONVERSION DECODER

17.4.1 BACKGROUND

Digital video broadcasting has had a major impact in both academic and industrial communities. A great deal of effort has been made to improve the coding efficiency at the transmission side and

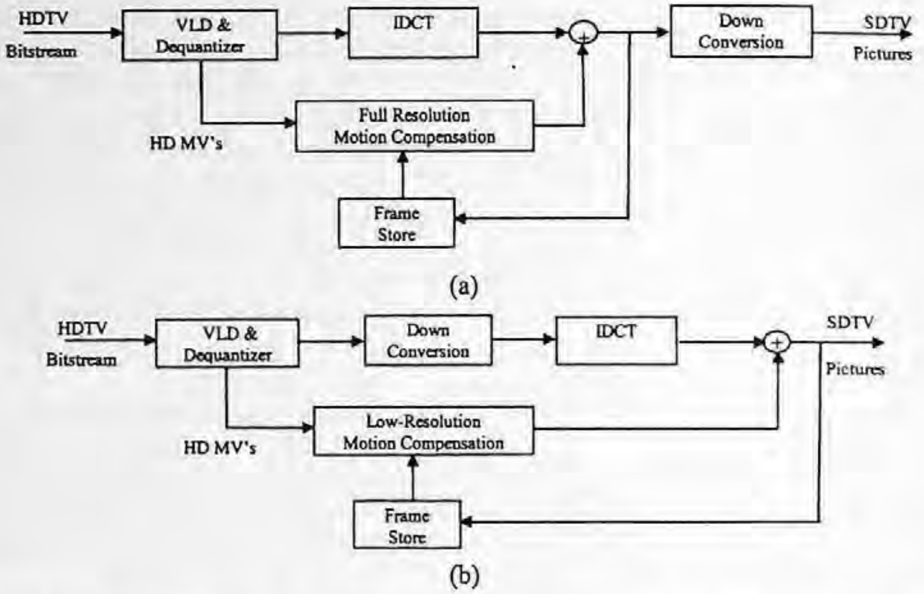


FIGURE 17.6 Decoder structures. (a) Block diagram of full-resolution decoder with down-conversion in the spatial domain. The quality of this output will serve as a drift-free reference. (b) Block diagram of low-resolution decoder. Down-conversion is performed within the decoding loop and is a frequency domain process. Motion compensation is performed from a low-resolution reference using motion vectors that are derived from the full-resolution encoder. Motion compensation is a spatial domain process.

offer cost-effective implementations in the overall end-to-end system. Along these lines, the notion of format conversion is becoming increasingly popular. On the transmission side, there are a number of different formats that are likely candidates for digital video broadcast. These formats vary in horizontal, vertical, and temporal resolution. Similarly, on the receiving side, there are a variety of display devices that the receiver should account for. In this section, we are interested in the specific problem of how to receive an HDTV bitstream and display it at a lower spatial resolution. In the conventional method of obtaining a low-resolution image sequence, the HD bitstream is fully decoded; then it is simply prefiltered and subsampled (ISO/IEC, 1993). The block diagram of this system is shown in Figure 17.6(a); it will be referred to as a full-resolution decoder (FRD) with spatial down-conversion. Although the quality is very good, the cost is quite high due to the large memory requirements. As a result, low-resolution decoders (LRDs) have been proposed to reduce some of the costs (Ng, 1993; Sun, 1993; Boyce et al., 1995; Bao et al., 1996). Although the quality of the picture will be compromised, significant reductions in the amount of memory can be realized; the block diagram for this system is shown in Figure 17.6(b). Here, incoming blocks are subject to down-conversion filters within the decoding loop. In this way, the down-converted blocks are stored into memory rather than the full-resolution blocks. To achieve a high-quality output with the low-resolution decoder, it is important to take special care in the algorithms for down-conversion and motion compensation (MC). These two processes are of major importance to the decoder as they have significant impact on the final quality. Although a moderate amount of complexity within the decoding loop is added, the reductions in external memory are expected to provide significant cost savings, provided that these algorithms can be incorporated into the typical decoder structure in a seamless way.

As stated above, the filters used to perform the down-conversion are an integral part of the low-resolution decoder. In Figure 17.6(b), the down-conversion is shown to take place before the IDCT. Although the filtering is not required to take place in the DCT domain, we initially assume that it takes place before the adder. In any case, it is usually more intuitive to derive a down-conversion filter in the frequency domain rather than in the spatial domain; this has been described by Pang et al. (1996), Merhav and Bhaskaran (1997), and Mokry and Anastassiou (1994). The major drawback of these approaches is that high frequency data is lost or not preserved very well. To overcome this, a method of down-conversion, which better preserves high-frequency data within the macroblock has been reported by Bao et al. (1996), Vetro and Sun (1998a); this method is referred to as frequency synthesis.

Although the above statement of the problem has only mentioned filtering-based approaches to memory reduction within the decoding loop, readers should be aware that other techniques have also been proposed. For the most part, these approaches rely on methods of embedded compression. For instance, de With et al. (1998) quantized the data being written to memory adaptively using a block predictive coding scheme; then a segment of macroblocks is fit into a fixed length packet. Similarly, Yu et al. (1999) proposed an adaptive min-max quantizer and edge detector. With this method, each macroblock is compressed to a fixed size to simplify memory access. Another, simpler approach may be to truncate the 8-bit data to 7 or 6 bits. However, in this case, it is expected the drift would accumulate very fast and result in poor reconstruction quality. Bruni et al. (1998) used a vectors quantization method, and Lei (1999) described a wavelet-based approach. Overall, these approaches offer exceptional techniques to reduce the memory requirements, but in most cases the reconstructed video would still be a high-resolution signal. The reason is that compressed high-resolution data are stored in memory rather than the raw, low-resolution data. For this reason, the remainder of this section emphasizes the filtering-based approach, in which the data stored in memory represent the actual low-resolution picture data.

The main novelty of the system that we describe is the filtering which is used to perform motion compensation from low-resolution anchor frames. It is well known that prediction drift has been difficult to avoid. It is partly due to the loss of high-frequency data from the down-conversion and partly due to the inability to recover the lost information. Although prediction drift cannot be totally avoided in a low-resolution decoder, it is possible to reduce the effects of drift significantly in contrast to simple interpolation methods. The solution that we described is optimal in the least-squares sense and is dependent on the method of down-conversion that is used (Vetro and Sun, 1998b). In its direct form, the solution cannot be readily applied to a practical decoding scheme. However, it is shown that a cascaded realization is easily implemented into the FRD-type structure (Vetro et al., 1998).

17.4.2 Frequency Synthesis Down-Conversion

The concept of frequency synthesis was first reported by Bao et al. (1996) and later expanded upon by Vetro and Sun (1998b). The basic premise is to better preserve the frequency characteristics of a macroblock in comparison to simpler methods which extract or cut specified frequency components of an 8 × 8 block. To accomplish this, the four blocks of a macroblock are subject to a global transformation — this transformation is referred to as frequency synthesis. Essentially, a single-frequency domain block can be realized using the information in the entire macroblock. From this, lower-resolution blocks can be achieved by cutting out the low-order frequency components of the synthesized block — this action represents the down-conversion process and is generally represented in the following way:

$$\tilde{\underline{A}} = X\underline{A}, \tag{17.11}$$

where $\underline{A}$ denotes the original DCT macroblock, $\underline{A}$ denotes the down-converted DCT block, and X is a matrix which contains the frequency synthesis coefficients. The original idea for frequency synthesis down-conversion was to extract an 8×8 block directly from the 16×16 synthesized block in the DCT domain as shown in Figure 17.7(a). The advantage of doing this is that the down-converted DCT block is directly applicable to an 8 × 8 IDCT (for which fast algorithms exist). The major drawback with regard to computation is that each frequency component in the synthesized

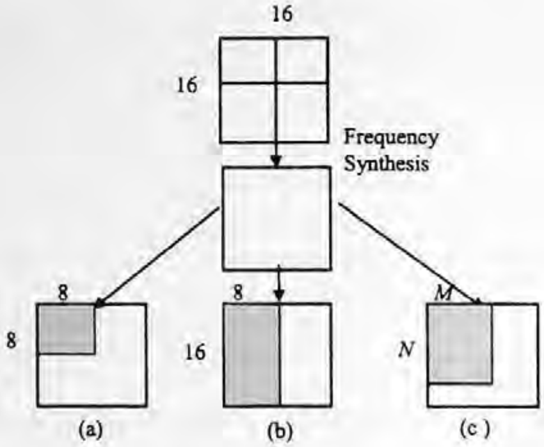


FIGURE 17.7 Concept of frequency synthesis down-conversion: (a) 256-tap filter applied to every frequency component to achieve vertical and horizontal down-conversion by a factor of two frame-based filtering; (b) 16-tap filter applied to frequency components in the same row to achieve horizontal down-conversion by two, picture structure is irrelevant; (c) illustration that the amount of synthesized frequency components which are retained is arbitrary.

block is dependent on all of the frequency components in each of the 8 × 8 blocks, i.e., each synthesized frequency component is the result of a 256-tap filter. The major drawback with regard to quality is that interlaced video with field-based predictions should not be subject to frame-based filtering (Vetro and Sun, 1998b). If frame-based filtering is used, it becomes impossible to recover the appropriate field-based data that is required to make field-based predictions. In areas of large motion, severe blocking artifacts will result.

Obviously, the original approach would incur too much computation and quality degradation, so, instead, the operations are performed separately and vertical down-conversion is performed on a field basis. In Figure 17.7(b), it is shown that a horizontal-only down-conversion can be performed. To perform this operation, a 16-tap filter is ultimately required. In this way, only the relevant row information is applied as the input to the horizontal filtering operation and the structure of the incoming video has no bearing on the down-conversion process. The reason is that the data in each row of a macroblock belong to the same field; hence the format of the output block will be unchanged. It is noteworthy that the set of filter coefficients is dependent on the particular output frequency index. For 1-D filtering, this means that the filters used to compute the second output index, for example, are different from those used to compute the fifth output index. Similar to the horizontal down-conversion, vertical down-conversion can also be applied as a separate process. As reasoned earlier, field-based filtering is necessary for interlaced video with field-based predictions.

However, since a macroblock consists of eight lines for the even field and eight lines for the odd field, and the vertical block unit is 8, frequency synthesis cannot be applied. Frequency synthesis is a global transformation and is only applicable when one wishes to observe the frequency characteristics over a larger range of data than the basic unit. Therefore, to perform the vertical down-conversion, we can simply cut the low-order frequency components in the vertical direction. This loss that we accept in the vertical direction is justified by the ability to perform accurate low-resolution MC that is free from severe blocking artifacts.

In the above, we have explained how the original idea to extract an 8 × 8 DCT block is broken down into separable operations. However, since frequency synthesis provides an expression for every frequency component in the new 16 × 16 block, it makes sense to generalize the down-conversion process so that decimation, which are multiples of $1/16$ can be performed. In Figure 17.7(c), an $M \times N$ block is extracted. Although this type of down-conversion filtering may not be appropriate before the IDCT operation and may not be appropriate for a bitstream containing

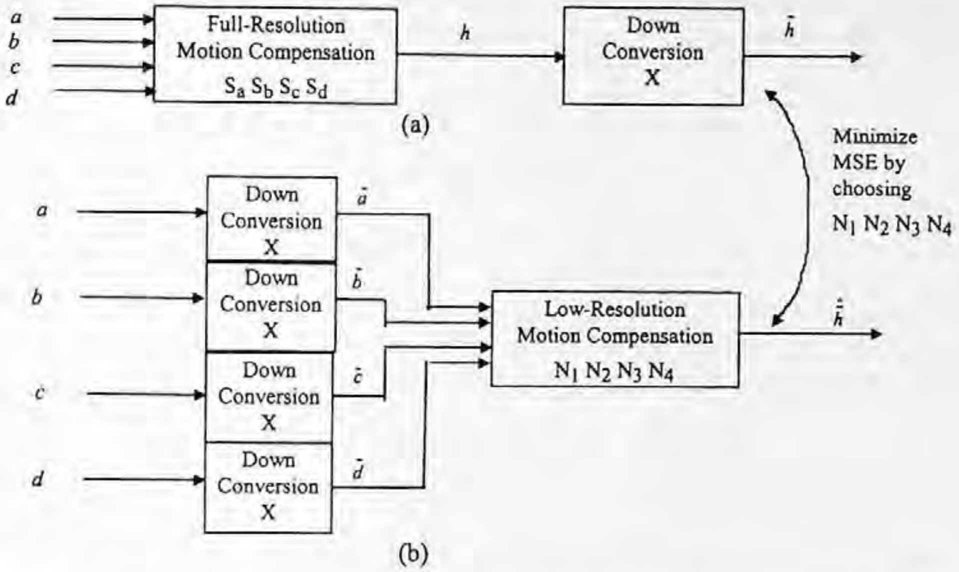


FIGURE 17.8 Comparison of decoding methods to achieve low-resolution image sequence. (a) FRD with spatial down-conversion; (b) LRD. The objective is to minimize the MSE between the two outputs by choosing N_1, N_2, N_3, and N_4 for a fixed down-conversion. (From Vetro, A. et al., *IEEE Trans. Consumer Elec.*, 44(3), 1998. With permission.)

field-based predictions, it may be applicable elsewhere, e.g., as a spatial domain filter somewhere else in the system and/or for progressive material. To obtain a set of spatial domain filters, an appropriate transformation can be applied. In this way, Equation 17.8 is expressed as

$$\underline{\tilde{a}} = x\underline{a}, \tag{17.12}$$

where the lowercase counterparts denote spatial equivalents. The expression which transforms X to x is derived in Appendix A, Section 17.4.6.

17.4.3 Low-Resolution Motion Compensation

The focus of this section is to provide an expression for the optimal set of low-resolution MC filters given a set of down-conversion filters. The resulting filters are optimal in the least-squares sense as they minimize the mean squared error (MSE) between a reference block and a block obtained through low-resolution MC. The results that have been derived by Vetro and Sun (1998a) assume that a spatial domain filter, x, is applied to incoming macroblocks to achieve the down-conversion. The scheme shown in Figure 17.8(a) illustrates the process by which reference blocks are obtained. First, full-resolution motion compensation is performed on macroblocks $\underline{a}$, $\underline{b}$, $\underline{c}$, and $\underline{d}$ to yield $\underline{h}$. To execute this process, the filters $S_a^{(r)}, S_b^{(r)}, S_c^{(r)}$, and $S_d^{(r)}$ are used. Basically, these filters represent the masking/averaging operations of the motion compensation in a matrix form. More on the composition of these filters can be found in Appendix B, Section 17.4.7. Once $\underline{h}$ is obtained, it is down-converted to $\underline{\tilde{h}}$ via the spatial filter, x:

$$\underline{\tilde{h}} = x\underline{h}. \tag{17.13}$$

The above block is considered to be the drift-free reference. On the other hand, in the scheme of Figure 17.8(b), the blocks $\underline{a}$, $\underline{b}$, $\underline{c}$, and $\underline{d}$ are first subject to the down-conversion filter, x, to yield the down-converted blocks, $\underline{\tilde{a}}$, $\underline{\tilde{b}}$, $\underline{\tilde{c}}$, and $\underline{\tilde{d}}$, respectively. By using these down-converted blocks as input to the low-resolution motion compensation process, the following expression can be assumed:

$$\hat{\underline{\tilde{h}}} = \begin{bmatrix} N_1 & N_2 & N_3 & N_4 \end{bmatrix} \begin{bmatrix} \underline{\tilde{a}} \\ \underline{\tilde{b}} \\ \underline{\tilde{c}} \\ \underline{\tilde{d}} \end{bmatrix}, \tag{17.14}$$

where N_k, $k = 1,2,3,4$ are the unknown filters which are assumed to perform the low-resolution motion compensation, and $\underline{\tilde{h}}$ is the low-resolution prediction. These filters are solved by differentiating the following objective function (Vetra and Sun, 1998a):

$$J\{N_k\} = \left\| \underline{\tilde{h}} - \hat{\underline{\tilde{h}}} \right\|^2, \tag{17.15}$$

with respect to each unknown filter and setting each result equal to zero. It can be verified that the optimal least-squares solution for these filters is given by

$$\begin{aligned} N_1^{(r)} &= xS_a^{(r)}x^+; \quad N_2^{(r)} = xS_b^{(r)}x^+ \\ N_3^{(r)} &= xS_c^{(r)}x^+; \quad N_4^{(r)} = xS_d^{(r)}x^+, \end{aligned} \tag{17.16}$$

where

$$x^+ = x^T \left(xx^T\right)^{-1} \tag{17.17}$$

is the Moore–Penrose inverse (Lancaster and Tismenetsky, 1985) for an $m \times n$ matrix with $m \le n$. In the solution of Equation 17.16, the superscript r is added to the filters, N_k, due to their dependency on the full-resolution motion compensation filters. In using these filters to perform the low-resolution motion compensation, the MSE between $\underline{\tilde{h}}$ and $\hat{\underline{\tilde{h}}}$ is minimized. It should be emphasized that Equation 17.16 represents a generalized set of MC filters which are applicable to any x, which operates on a single macroblock. For the special case of the 4 × 4 cut, these filters are equivalent to the ones that were determined by Morky and Anastassiou (1994) to minimize the drift.

In Figure 17.9, two equivalent MC schemes are shown. However, for implementation purposes, the optimal MC scheme is realized in a cascade form rather than a direct form. The reason is that the direct-form filters are dependent on the matrices, which perform full-resolution MC. Although, these matrices were very useful in analytically expressing the full-resolution MC process, they require a huge amount of storage due to their dependency on the prediction mode, motion vector, and half-pixel-accuracy. Instead, the three linear processes in Equation 17.13 are separated, so that an up-conversion, full-resolution MC, and down-conversion can be performed. Although one may be able to guess such a scheme, we have proved here that it is an optimal scheme provided the up-conversion filter is a Moore–Penrose inverse of the down-conversion filter. Vetro and Sun (1998b), the optimal MC scheme, which employs frequency synthesis, to a nonoptimal MC scheme, which employs bilinear interpolation, and an optimal MC scheme, which employs the 4 × 4 cut down-conversion. Significant reductions in the amount of drift were realized by both optimal MC schemes over the method, which used bilinear interpolation as the method of up-conversion. But more importantly, a 35% reduction in the amount of drift was realized by the optimal MC scheme using frequency synthesis over the optimal MC scheme using the 4 × 4 cut.

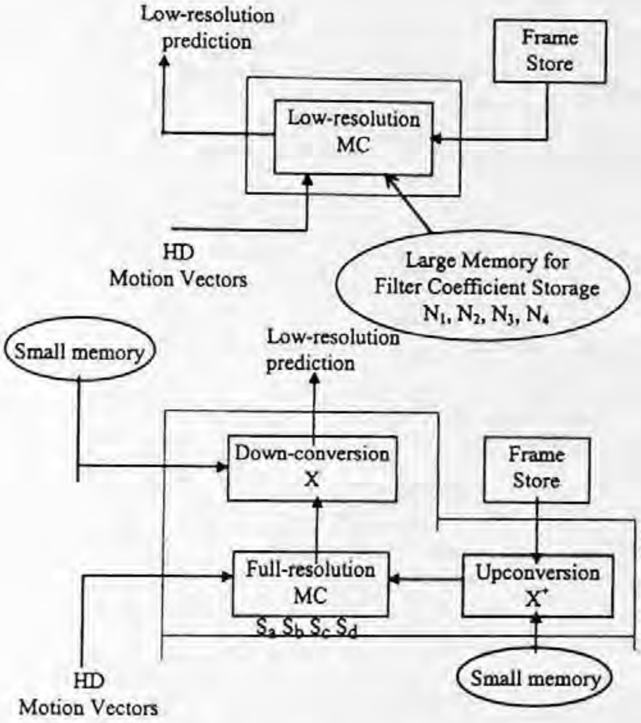
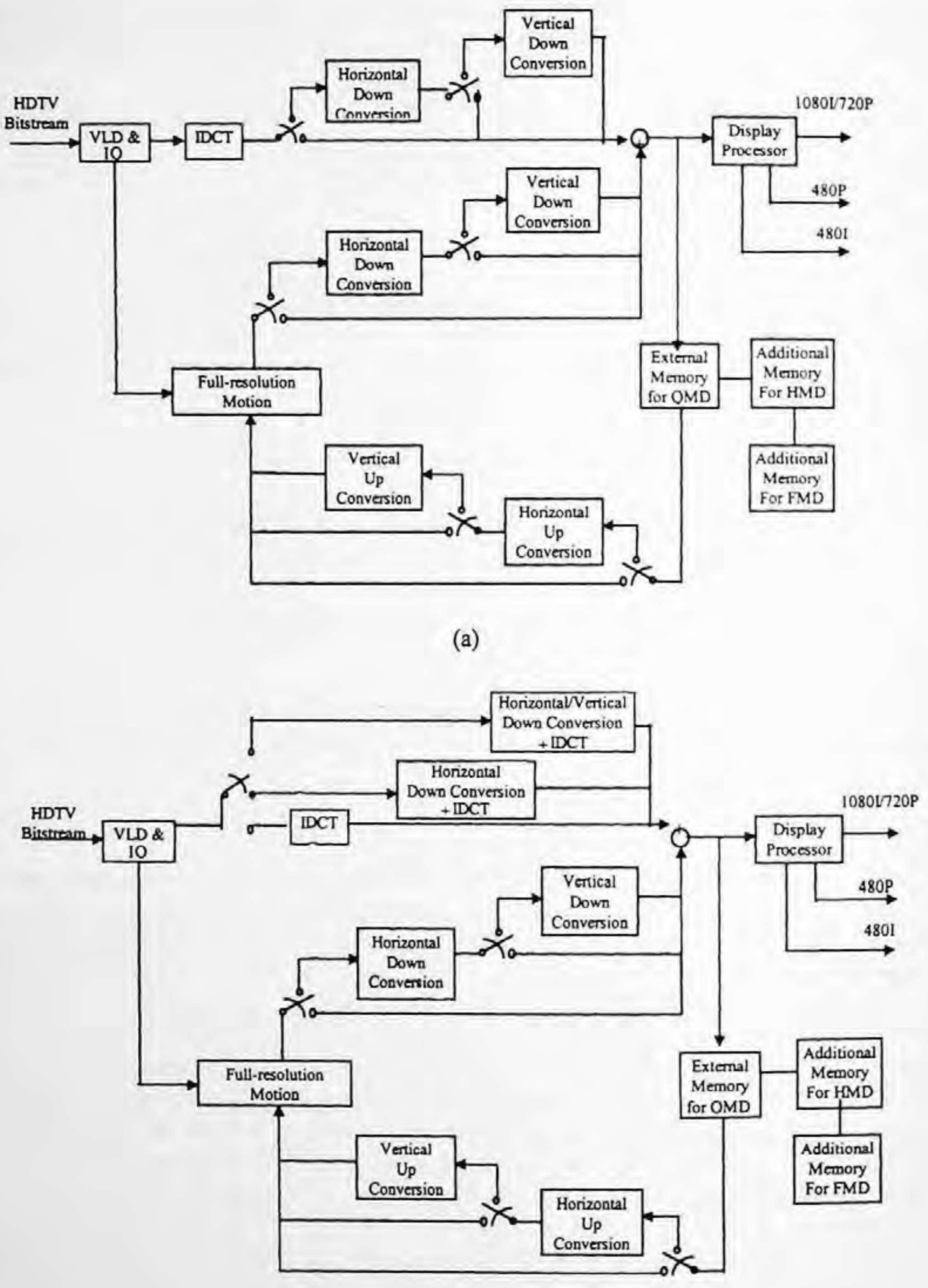


FIGURE 17.9 Optimal low-resolution MC scheme: direct form (top) vs. cascade form (bottom). Both forms yield equivalent quality, but vary significantly in the amount of internal memory. (From Vetra, A., et al., *IEEE Trans. Consumer Elec.*, 44(3), 1998. With permission.)

17.4.4 Three-Layer Scalable Decoder

In this section, we show how the key algorithms for down-conversion and motion compensation are integrated into a three-layer scalable decoder. The central concept of this decoder is that three layers of resolution can be decoded using a decreased amount of memory for the lower resolution layers. Also, regardless of which layer is being decoded, much of the logic can be shared. Three possible decoder configurations are considered: full-memory decoder (FMD), half-memory decoder (HMD), and quarter-memory decoder (QMD). The low-resolution decoder configurations are based on the key algorithms, which were described for down-conversion and motion compensation. In the following, three possible architectures are discussed that provide equal quality, but vary in system-level complexity. The first (ARCH1) is based on the low-resolution decoder modeled in Figure 17.6(b), the second (ARCH2) is very similar, but attempts to reduce the IDCT computation, while the third (ARCH3) is concerned with the amount of interface with an existing high-level decoder.

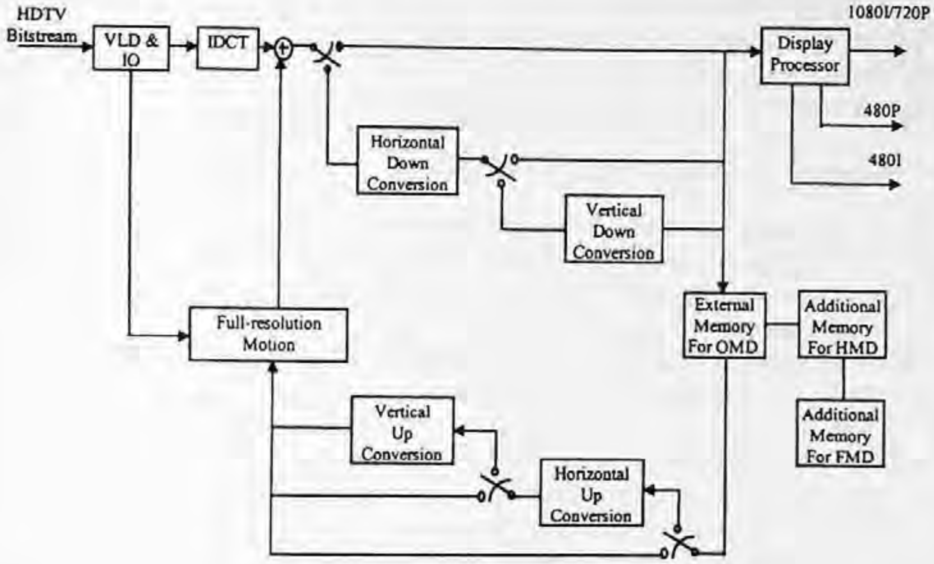
With regard to functionality, all of the architectures share similar characteristics. For one, an efficient implementation is achieved by arranging the logic in a hierarchical manner, i.e., employing separable processing. In this way, the FMD configuration is the simplest and serves as the logic core on which other decoder configurations are built. In the HMD configuration, an additional horizontal down-conversion and up-conversion are performed. In the QMD configuration, all of the logic components from the HMD are utilized, such that an additional vertical down-conversion is performed after a horizontal down-conversion, and an additional vertical up-conversion is performed after a horizontal up-conversion. In summary, the logic for the HMD is built on the logic for the FMD, and the logic for the QMD is built on the logic of the HMD. The total system contains a moderate increase in logic, but HD bitstreams may be decoded to a lower resolution with a smaller amount of external memory. By simply removing external memory, lower layers can be achieved at a reduced cost.



(a)

(b)

FIGURE 17.10 Block diagram of various three-layer scalable decoder architectures; all architectures provide equal quality with varying system complexity: (a) ARCH1, derived directly from block diagram of assumed low-resolution decoder; (b) ARCH2, reduce computation of IDCT by combining down-conversion and IDCT filters; (c) ARCH3, minimize interface with existing HL decoder by moving linear filtering for down-conversion outside of the adder. (From Vetro, A. et al., *IEEE Trans. Consumer Elec.*, 44(3), 1998. With permission.)



(c)

FIGURE 17.10 (continued)

The complete block diagram of ARCH1 is shown in Figure 17.10(a). The diagram shown here assumes two things: (1) the initial system model of a low-resolution decoder from Figure 17.6(b) is assumed, and (2) the down-conversions in the incoming branch are performed after the IDCT to avoid any confusion regarding macroblock format conversions in the DCT domain (Vetro and Sun, 1998b). In looking at the resulting system, it is evident that full computation of the IDCT is required, and that two independent down-conversion operations must be performed. The latter is necessary so that low-resolution predictions are added to low-resolution residuals. Overall, the increase in logic for the added feature of memory savings is quite small. However, it is evident that ARCH1 is not the most cost-effective implementation, but it represents the foundation of previous assumptions, and allows us to analyze better the impact of the two modified architectures to follow.

In Figure 17.10(b), the block diagram of ARCH2 is shown. In this system, realizing that the IDCT operation is simply a linear filter reduces the combined computation for the IDCT and down-conversion. In the FMD, we know that a fast IDCT is applied separately to the rows and columns of an 8 × 8 block. For the HMD, our goal is to combine the horizontal down-conversion with the horizontal IDCT. In the 1-D case, an 8 × 16 matrix can represent the horizontal down-conversion, and an 8 × 8 matrix can represent the horizontal IDCT. Combining these processes, such that the down-conversion operates on the incoming DCT rows first, results in a combined 8 × 16 matrix. To complete the transformation, the remaining columns can then be applied to the fast IDCT. In the above description, computational savings are achieved in two places: first, the horizontal IDCT is fully absorbed into the down-conversion computation which must take place anyway, and, second, the fast IDCT is utilized for a smaller amount of columns. In the case of the QMD, these same principles can be used to combine the vertical down-conversion with the vertical IDCT. In this case, one must be aware of the macroblock type (field-DCT or frame-DCT) so that an appropriate filter can be applied. In contrast to the previous two architectures, ARCH3 assumes that the entire front-end processing of the decoder is used; it is shown in Figure 17.5. In this way, the adder is always a full-resolution adder, whereas in ARCH1 and ARCH2, the adder needed to handle all three layers of resolution. The major benefit of ARCH3 is that it does not require many interfaces with the existing decoder structure. The memory is really the only place where a new interface needs to be defined. Essentially, a down-conversion filtering may be applied before storing the data, and an up-conversion filtering may be applied, as the data is needed for full-resolution MC. This final architecture is similar in principle to the embedded compression schemes that were mentioned in the beginning of this section. The main difference is that the resolution of the data is decreased rather than compressed. This allows a simpler means of low-resolution display.

17.4.5 Summary of Down-Conversion Decoder

A number of integrated solutions for a scalable decoder have been presented. Each decoder is capable of decoding directly to a lower resolution using a reduced amount of memory in comparison with the memory required by the high-level decoder. The method of frequency synthesis is successful in better preserving the high-frequency data within a macroblock, and the filtering that is used to perform optimal low-resolution MC is capable of minimizing the drift. It has been shown that a realizable implementation can be achieved, such that the filters for optimal low-resolution MC are equivalent to an up-conversion, full-resolution MC, and for down-conversion, where the up-conversion filters are determined by a Moore–Penrose inverse of the down-conversion. The amount of logic required by these processes is kept minimal since they are realized in a hierarchical structure. Since the down-conversion and up-conversion processes are linear, the architecture design is flexible in that equal quality can be achieved with varying levels of system complexity. The first architecture that we examined came from the initial assumptions that were made on the low-resolution decoder, i.e., a down-conversion is performed before the adder. It was noted that a full IDCT computation was required and that a down-conversion must be performed in two places. As a result, a second architecture was presented to reduce the IDCT computation, and a third was presented to minimize the amount of interface with the existing high-level decoder. The major point here is that the advantages of ARCH2 and ARCH3 cannot be realized by a single architecture. The reason is that performing a down-conversion in the incoming branch reduces the IDCT computation; therefore, a down-conversion must be performed after the full-resolution MC as well. In any case, equal quality is offered by each architecture and the quality is of commercial grade.

17.4.6 DCT-to-Spatial Transformation

Our objective in this section is to express the following DCT domain relationship:

$$\tilde{A}(k,l)=\sum_{p=0}^{M-1}\sum_{q=0}^{N-1}\left[X_{k,l}(p,q)A(p,q)\right] \tag{17.18}$$

as

$$\tilde{a}(i,j)=\sum_{s=0}^{M-1}\sum_{t=0}^{N-1}\left[X_{i,j}(s,t)a(s,t)\right], \tag{17.19}$$

where $\tilde{A}$ and $\tilde{a}$ are the DCT and spatial output, A and a are the DCT and spatial input, and X and x are the DCT and spatial filters, respectively. By definition, the $M \times N$ DCT transform is defined by

$$A(k,l)=\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}a(i,j)\psi_k^M(i)\psi_l^N(j) \tag{17.20}$$

and its inverse, the $M \times N$ IDCT by

$$a(i,j)=\sum_{k=0}^{M-1}\sum_{l=0}^{N-1}A(k,l)\psi_k^M(i)\psi_l^N(j), \tag{17.21}$$

where the basis function is given by

$$\psi_k^N = \sqrt{\frac{2}{N}}\,\alpha(k)\cos\left(\frac{2i+1}{2N}k\pi\right) \tag{17.22}$$

and

$$\alpha(k) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for} \quad k = 0; \\ 1 & \text{for} \quad k \neq 0. \end{cases} \tag{17.23}$$

By substituting Equation 17.22 into the expression for the IDCT yields

$$\begin{aligned} \tilde{a}(i,j) &= \sum_{k=0}^{M-1}\sum_{l=0}^{N-1}\psi_k^M(i)\psi_l^N(j)\cdot\left[\sum_{p=0}^{M-1}\sum_{q=0}^{N-1}X_{k,l}(p.q)A(p.q)\right] \\ &= \sum_{p=0}^{M-1}\sum_{q=0}^{N-1}A(p,q)\cdot\left[\sum_{k=0}^{M-1}\sum_{l=0}^{N-1}X_{k,l}(p,q)\psi_k^M(i)\psi_l^N(j)\right]. \end{aligned} \tag{17.24}$$

Substituting the DCT definition into the above gives the following,

$$\bar{a}(i,j) = \sum_{p=0}^{M-1}\sum_{q=0}^{N-1}\left[\sum_{s=0}^{M-1}\sum_{t=0}^{N-1}a(s,t)\psi_p^M(s)\psi_q^N(t)\right]\sum_{k=0}^{M-1}\sum_{l=0}^{N-1}\left[X_{k,l}(p,q)\cdot\psi_p^M(i)\psi_q^N(j)\right]. \tag{17.25}$$

Finally, Equation 17.17 can be formed with

$$x_{i,j}(s,t) = \sum_{k=0}^{M-1}\sum_{l=0}^{N-1}\psi_k^M(i)\cdot\psi_l^N(j)\left[\sum_{k=0}^{M-1}\sum_{l=0}^{N-1}\left(X_{k,l}(p,q)\cdot\psi_p^M(s)\psi_q^N(t)\right]\right. \tag{17.26}$$

and the transformation is fully defined.

17.4.7 Full-Resolution Motion Compensation in Matrix Form

In 2-D, a motion compensated macroblock may have contributions from at most four macroblocks per motion vector. As noted in Figure 17.11, macroblocks *a*, *b*, *c*, and *d* include four 8 × 8 blocks each. These subblocks are raster-scanned so that each macroblock can be represented as a vector. According to the motion vector, (*dx*, *dy*), a local reference, (y_1,y_2), is computed to indicate where the origin of the motion compensated block is located; the local reference is determined by

$$\begin{aligned} y_1 &= dy - 16\cdot\left[Integer(dy/16) - \gamma(dy)\right] \\ y_2 &= dx - 16\cdot\left[Integer(dx/16) - \gamma(dx)\right], \end{aligned} \tag{17.27}$$

where

$$\gamma(d) = \begin{cases} 1, & \text{if} \quad d < 0 \text{ and } d \bmod 16 = 0 \\ 0, & \text{otherwise.} \end{cases} \tag{17.28}$$

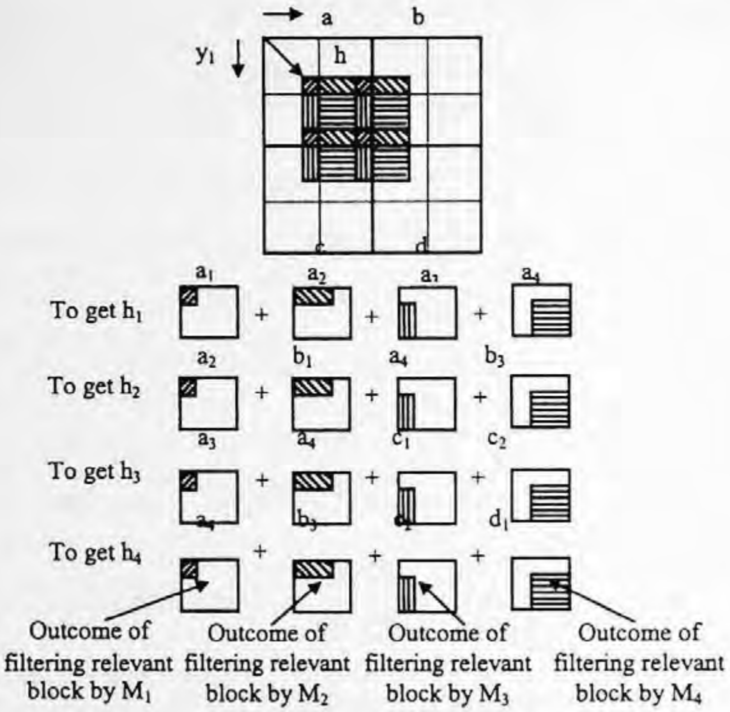

a
b
y1
h
c
d
a1
a2
a3
a4
To get h1
a2
b1
a4
b3
To get h2
a3
a4
c1
c2
To get h3
a4
b3
c2
d1
To get h4
Outcome of filtering relevant block by M1
Outcome of filtering relevant block by M2
Outcome of filtering relevant block by M3
Outcome of filtering relevant block by M4

FIGURE 17.11 Relationship between the input and output blocks of the motion compensation process in the FRD. (From Vetro, A. et al., *IEEE Trans. Consumer Elec.*, 44(3), 1998. With permission.)

The reference point for this value is the origin of the upper-left-most input macroblock. With this, the motion-compensated prediction may be expressed as

$$\underline{h} = \begin{bmatrix}\underline{h_1}\\ \underline{h_2}\\ \underline{h_3}\\ \underline{h_4}\end{bmatrix} = \begin{bmatrix} S_a^{(r)} & S_b^{(r)} & S_c^{(r)} & S_d^{(r)} \end{bmatrix} \cdot \begin{bmatrix}\underline{a}\\ \underline{b}\\ \underline{c}\\ \underline{d}\end{bmatrix}; \quad r = 1, 2, 3, 4. \tag{17.29}$$

As an example, Figure 17.11 considers $(y_1, y_2) \in [0,7]$, which implies that $r = 1$. In this case the motion compensation filters are given by

$$\begin{gathered} S_a^{(1)} = \begin{bmatrix} M_1 & M_2 & M_3 & M_4 \\ 0 & M_1 & 0 & M_3 \\ 0 & 0 & M_1 & M_2 \\ 0 & 0 & 0 & M_1 \end{bmatrix}, \quad S_b^{(1)} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ M_2 & 0 & M_4 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & M_2 & 0 \end{bmatrix}, \\ S_c^{(1)} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ M_3 & M_4 & 0 & 0 \\ 0 & M_3 & 0 & 0 \end{bmatrix}, \quad S_c^{(1)} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ M_4 & 0 & 0 & 0 \end{bmatrix}. \end{gathered} \tag{17.30}$$

In the above equations, the M_1, M_2, M_3, and M_4 matrices operate on the relevant 8 × 8 blocks of *a*, *b*, *c*, and *d*. Their elements will vary according to the amount of overlap as indicated by (y_1, y_2) and the type of prediction. The type of prediction may be frame based or field based and is predicted with half-pixel accuracy. As a result, the matrices $S_a^{(r)}$, $S_b^{(r)}$, $S_c^{(r)}$, and $S_d^{(r)}$, are extremely sparse and may only contain nonzero values of 1, ½, and ¼. For different values of (y_1,y_2) the configuration of the above matrices will change: $y_1 \in [0,7]$ and $y_2 \in [8,15]$ implies $r = 2$; $y_1 \in [8,15]$ and $y_2 \in [0,7]$ implies $r = 3$; $y_1, y_2 \in [8,15]$ implies $r = 4$. The resulting matrices can easily be formed using the concepts illustrated in Figure 17.11.

17.5 ERROR CONCEALMENT

17.5.1 BACKGROUND

Practical communications channels available for delivery of compressed digital video are characterized by occasional bit error and/or packet loss, although the actual impairment mechanism varies widely with the specific medium under consideration. The class of decoder error concealment schemes described here is based on identification and predictive replacement of picture regions affected by bit error or data loss. It is noted that this approach is based on conversion (via appropriate error/loss detection mechanisms) of the transmission medium into an erasure channel in which all error or loss events can be identified in the received bit-stream. In a block-structured compression algorithm such as MPEG, all channel impairments are manifested as erasures of video units (such as MPEG macroblocks or slices). Concealment at the decoder is then based on exploiting temporal and spatial picture redundancy to obtain an estimate of erased picture areas. The efficiency of error concealment depends on redundancies in pictures and on redundancies in the compressed bitstream that are not removed by source coding. Block compression algorithms do not remove a considerable amount of inter-block redundancies, such as structure, texture, and motion information about objects in the scene.

To be more specific, error resilience for compressed video can be achieved through the addition of suitable transport and error concealment methods, as outlined in the system block diagram shown in Figure 17.12.

The key elements of such a robust video delivery system are outlined below:

- The video signal is encoded using an appropriate video compression syntax such as MPEG. Note that we have restricted consideration primarily to the practical case in which the video compression process itself is not modified, and robustness is achieved through additive transport and decoder concealment mechanisms (except for I-frame motion described in Section 17.4.3). This approach simplifies encoder design, since it separates

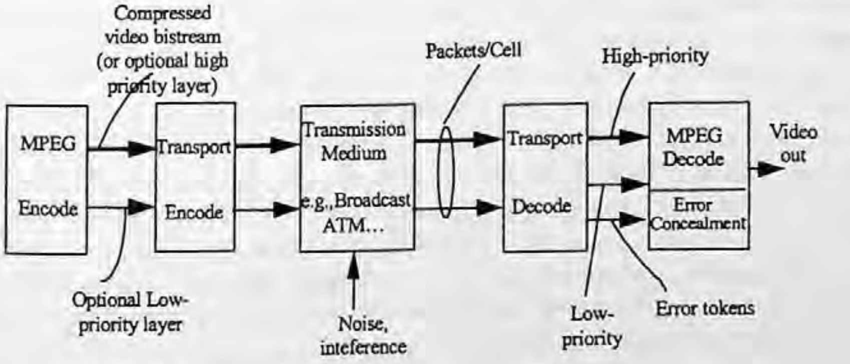


FIGURE 17.12 System block diagram of visual communication system.

media-independent video compression functions from media-dependent transport operations. On the receiver side, although a similar separation is substantially maintained, the video decoder must be modified to support an "error token" interface and error concealment functionality.

- Compressed video data is organized into a systematic data structure with appropriate headers for identification of the temporal and spatial pixel-domain location of encoded data (Joseph et al., 1992b). When an erroneous/lost packet is detected, these video units serve as resynchronization points for resumption of normal decoding, while the headers provide a means for precisely locating regions of the picture that were not correctly received. Note that two-tier systems may require additional transport-level support for high- and low-priority (HP/LP) resynchronization (Siracusa et al., 1993).
- The video bitstream may optionally be segregated into two layers for prioritized transport (Ghanbari, 1989; Kishno et al., 1989; Karlsson and Vetterli, 1989; Zdepski et al., 1989; Joseph et al., 1992a,b; Siracusa, 1993) when a high degree of error resilience is required. Note that separation into high and low priorities may be achieved either by using a hierarchical (layered) compression algorithm (Ghanbari, 1989; Siracusa, 1993) or by direct codeword parsing (Zdepski et al., 1989; 1990). Note that both these layering mechanisms have been accepted for standardization by MPEG-2 (ISO/IEC, 1995).
- Once the temporal and spatial location(s) corresponding to lost or incorrectly received packets is determined by the decoder, it will execute an error-concealment procedure for replacement of lost picture areas with subjectively acceptable material estimated from available picture regions (Harthanck et al., 1986; Jeng and Lee, 1991; Wang and Zhu, 1991). Generally, this error concealment procedure will be applied to all erased blocks in one-tier (single-priority) transmission systems, while for two-tier (HP/LP) channels the concealment process may optionally ignore loss of LP data.
- In the following subsections, the technical detail of some commonly used error concealment algorithms is provided. Specifically, we focus on the recovery of codeword errors and errors that affect the pixels within a macroblock.

17.5.2 Error Concealment Algorithms

In general, design of specific error-concealment strategies depends on the system design. For example, if two-layered transmission is used, the receiver should be designed to conceal high-priority error and low-priority error with different strategies. Moreover, if some redundancy ("steering information") could be added to the encoder the concealment could be more efficient. However, we first assume that the encoder is defined for maximum compression efficiency, and that concealment is only performed in the receiver. It should be noted that some exemptions exist for this assumption. These exemptions include the use of I-frame motion vectors, scalability concealment, and limitation of slice length (to perform acceptable concealment in the pixel domain the limitation of slice length exists, i.e., the length of slices cannot be longer than one row of picture). Figure 17.13 shows a block diagram of a generic one/two-tier video decoder with error concealment.

Note that the figure shows two stages of decoder concealment in the codeword domain and pixel domain, respectively. Codeword domain concealment, in which locally generated decodable codewords (e.g., B-picture motion vectors, end-of-block code, etc.) are inserted into the bitstream, is convenient for implementation of simple temporal replacement functions (which in principle can also be performed in the pixel domain). The second stage of pixel domain processing is for temporal and spatial operations not conveniently done in the codeword domain. Advanced spatial processing will generally have to be performed in the pixel domain, although limited codeword domain options can also be identified.

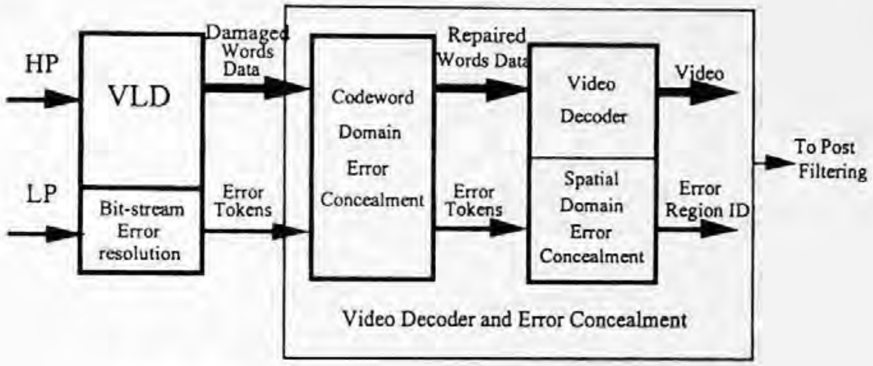

HP
LP
VLD
Bit-stream Error resolution
Damaged Words Data
Error Tokens
Codeword Domain Error Concealment
Repaired Words Data
Error Tokens
Video Decoder
Spatial Domain Error Concealment
Video
Error Region ID
To Post Filtering
Video Decoder and Error Concealment

FIGURE 17.13 MPEG video decoder with error concealment.

17.5.2.1 Codeword Domain Error Concealment

The codeword domain concealment receives video data and error tokens from the transport processor/VLD. Under normal conditions, no action is taken and the data are passed along to the video decoder. When an error token is received, damaged data are repaired to the extent possible by insertion of locally generated codewords and resynchronization codes. An error region ID is also created to indicate the image region to be concealed by subsequent pixel domain processing. Two mechanisms have been used in codeword domain error concealment: neglect the effect of lost data by declaring an end of block (EOB), or replace the lost data with a pseudo-code to handle the macroblock-types or other VLC codes. If high-level data such as dc or macroblock header is lost, the codeword domain concealment with pseudo-codes can only provide signal resynchronization (decodability) and replaces the image scene with a fixed gray level in the error region. Obviously, further improvement is needed in the video decoder. This task is implemented with the error concealment in the video decoder. It is desirable to replace erased I- or P-picture regions with a reasonably accurate estimate to minimize the impact of frame-to-frame propagation.

17.5.2.2 Spatiotemporal Error Concealment

In general, two basic approaches are used for spatial domain error concealment: temporal replacement and spatial interpolation. In temporal replacement, as shown in Figure 17.14, the spatially corresponding ones in the previously decoded data with motion compensation replace the damaged blocks in the current frame if motion information is available. This method exploits temporal redundancy in the reconstructed video signals and provides satisfactory results in areas with small

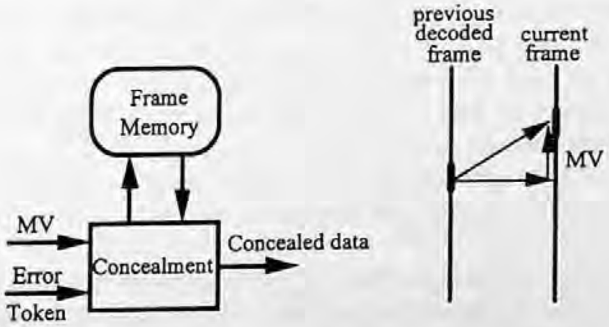


FIGURE 17.14 Error concealment uses temporal replenishment with motion compensation.

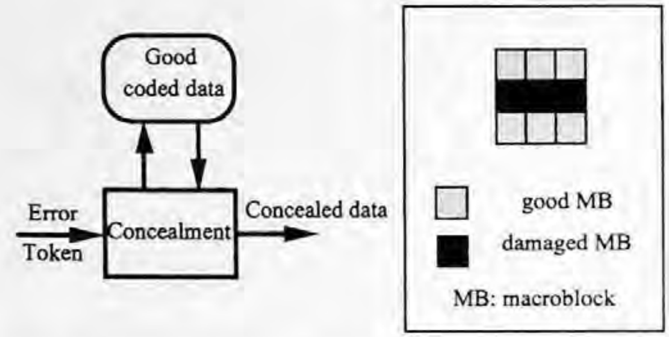


FIGURE 17.15 Error concealment uses spatial interpolation with the data from good neighbors. (From Sun, H. and Kwok, W. *IEEE Trans. Image Proc.*, 4(4), 470–477, 1995. With permission.)

motion and for which motion vectors are provided. If motion information is lost, this method will fail in the moving areas. In the method of spatial interpolation as shown in Figure 17.15, the lost blocks are interpolated by the data from the adjacent nonerror blocks with maximally smooth reconstruction criteria or other techniques.

In this method, the correlation between adjacent blocks in the received and reconstructed video signals is exploited. However, severe blurring will result from this method if data in adjacent blocks are also lost. In an MPEG decoder, temporal replacement outlined above is based on previously decoded anchor (I, P) pictures that are available in the frame memory. If motion vectors corresponding to pixels in the erasure region can also be estimated, this temporal replacement operation can be improved via motion compensation. Also, in the MPEG decoder, groups of video pixels (blocks, macroblocks, or slices) are separately decoded, so that pixel values and motion information corresponding to adjacent picture regions are generally available for spatial concealment. However, estimation from horizontally adjacent blocks may not always be useful since cell loss tends to affect a number of adjacent blocks (due to the MPEG and ATM data structures); also differential encoding between horizontally adjacent blocks tends to limit the utility of data obtained from such neighbors. Therefore, most of the usable spatial information will be located in blocks above or below the damaged region. That is, vertical processing/concealment is found to be most useful due to the transmission order of the data.

For I-pictures, the damaged data can be reconstructed by either temporal replacement from the previously decoded anchor frame or by spatial interpolation from good neighbors. These two methods will be discussed later. For P- and B-pictures, the main strategy to conceal the lost data is to replace the region with pixels from the corresponding (and possibly motion-compensated) location in the previously decoded anchor. In this replacement the motion vectors play a very important role. In other words, if "good" estimates of motion information can be obtained, its use may be the least noticeable correction. Since DPCM coding for motion vectors only exploited the correlations between the horizontally neighboring macroblocks, the redundancy between the vertical neighborhood still exists after encoding. Therefore, the lost motion information can be estimated from the vertical neighbors. In the following, three algorithms that have been developed for error concealment in the video decoder are described.

Algorithm 1: Spatial interpolation of missing I-picture data and temporal replacement for P- and B-pictures with motion compensation (Sun et al., 1992a):

For I-pictures, dc values of damaged block are replaced by the interpolation from the closest top and bottom good neighbors; the ac coefficients of those blocks are synthesized from the dc values of the surrounding neighboring blocks.

For P-pictures, the previously decoded anchor frames with motion compensation replace the lost blocks. The lost motion vectors are estimated by interpolation of the ones from the top and bottom macroblocks. If motion vectors in both top and bottom macroblocks are not available, zero motion vectors are used. The same strategy is used for B-pictures; the only difference is that the closest anchor frame is used. In other words, the damaged part of the B-picture could be replaced by either the forward or backward anchor frame, depending on its temporal position.

Algorithm 2: Temporal replacement of missing I-picture data and temporal replacement for P- and B-pictures with top motion compensation:

For I-pictures, the damaged blocks are replaced with the colocated ones in the previously decoded anchor frame.

For P- and B-pictures, the closest previously decoded anchor frame replaces the damaged part with motion compensation as in the Algorithm 1. The only difference is that the motion vectors are estimated only from the closest top macroblock instead of interpolation of top and bottom motion vectors. This makes the implementation of this scheme much easier. If these motion vectors are not available, then zero motion vectors are used.

In the above two algorithms, the damaged blocks in an I-picture (anchor frame) are concealed by two methods: temporal replacement and spatial interpolation. Temporal replacement is able to provide high-resolution image data to substitute for lost data; however, in motion areas, a big difference might exist between the current intracoded frame and the previously decoded frame. In this case, temporal replacement will produce large shearing distortion unless some motion-based processing can be applied at the decoder. However, this type of processing is not generally available since it is a computationally demanding task to compute motion trajectories locally at the decoder. In contrast, the spatial interpolation approach synthesizes lost data from the adjacent blocks in the same frame. Therefore, the intraframe redundancy between blocks is exploited, while the potential problem of severe blurring due to insufficient high-order ac coefficients for active areas. To alleviate this problem, an adaptive concealment strategy can be used as a compromise; this is described in Algorithm 3.

Algorithm 3: Adaptive spatiotemporal replacement of missing I-picture data and temporal replacement with motion compensation for P- and B-pictures:

For I-pictures, the damaged blocks are concealed with temporal replacement or spatial interpolation according to the decision made by the top and bottom macroblocks, which is shown in Figure 17.16. The decision of which concealment method to use will be based on the more cheaply obtained measures of image activity from the neighboring top and bottom macroblocks. One candidate for the decision processor is to make the decision based on prediction error statistics measured in the neighborhood. The decision region is shown in Figure 17.16, where

$$\begin{aligned} \text{VAR} &= E\left[(x-\hat{x})^2\right], \\ \text{VAROR} &= E\left[x^2\right]-\mu^2, \end{aligned} \tag{17.31}$$

and x is the neighboring good macroblock data, $\hat{x}$ is the data of the corresponding macroblock in the previously decoded frame at the colocated position, and μ is the average value of the neighboring good macroblock data in the current frame. One can appreciate that VAR is indicative of the local motion and VAROR of the local spatial detail. If VAR > VAROR and VAR > T, where T is a preset threshold value which is set to 5 in the experiments, the concealment method is spatial interpolation; if VAR < VAROR or VAR < T, the concealment method is temporal replacement.

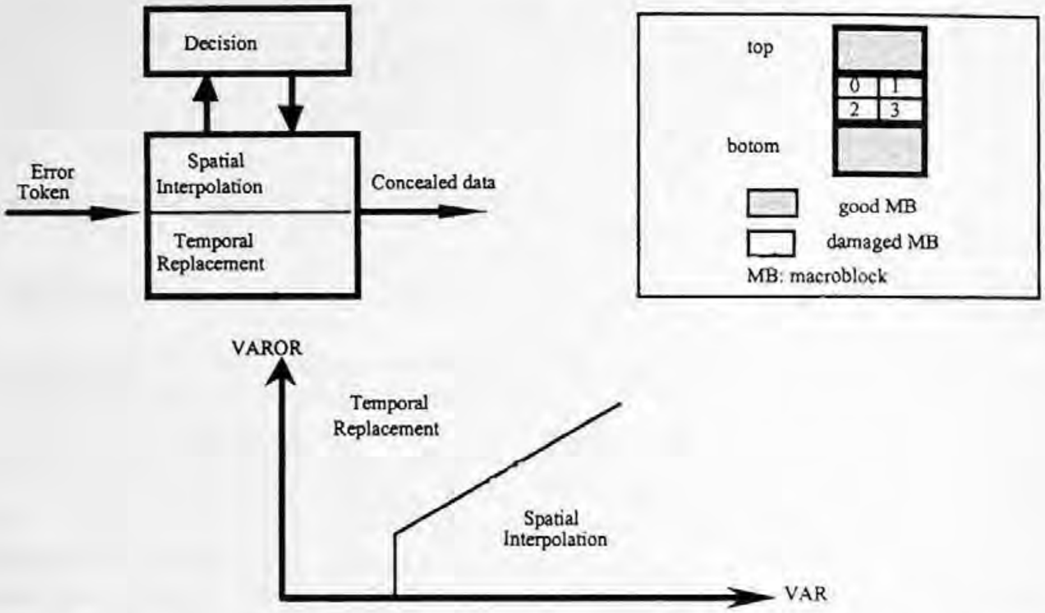


FIGURE 17.16 Adaptive error concealment strategy. (From Sun, H. and Kwok, W., *IEEE Trans. Image Proc.*, 4(4), 470–477, 1995. With permission.)

It should be noted that the concealment for luminance is performed on a block basis instead of macroblock basis, while the chrominance is still on the macroblock basis. The detailed decisions for the luminance blocks are described as follows:

- If both top and bottom are temporally replaced, then four blocks (0, 1, 2, and 3) are replaced by the colocated ones (colocated means no motion compensation) in the previously decoded frame.
- If top is temporally replaced and bottom is spatially interpolated, then blocks 0 and 1 are replaced by the colocated ones in the previously decoded anchor frame and blocks 2 and 3 are interpolated from the block boundaries.
- If top is spatially interpolated and bottom is temporally replaced, then blocks 0 and 1 are interpolated from the boundaries, and blocks 2 and 3 are replaced by the colocated ones in the previously decoded anchor frame.
- If both top and bottom are not temporally replaced, all four blocks are spatially interpolated.

In spatial interpolation, a maximal smoothing technique with boundary conditions under certain smoothness measures is used. The spatial interpolation process is carried out with two steps: the mean value of the damaged block is first bilinearly interpolated with ones from the neighboring blocks; then spatial interpolation for each pixel is performed with a Laplacian operator. Minimizing the Laplacian on the boundary pixels using the iterative process (Wang and Zhu, 1991) enforces the process of maximum smoothness.

For P- and B-pictures a similar concealment method is used as in Algorithm 2 except motion vectors from top and bottom neighboring macroblocks are used for top two blocks and bottom two blocks, respectively.

A schematic block diagram for implementation of adaptive error concealment for intracoded frames is given in Figure 17.17. Corrupted macroblocks are first indicated by error tokens obtained via the transport interface. Then, a decision regarding which concealment method (temporal replacement or spatial interpolation) should be used is based on easily obtained measures of image activity

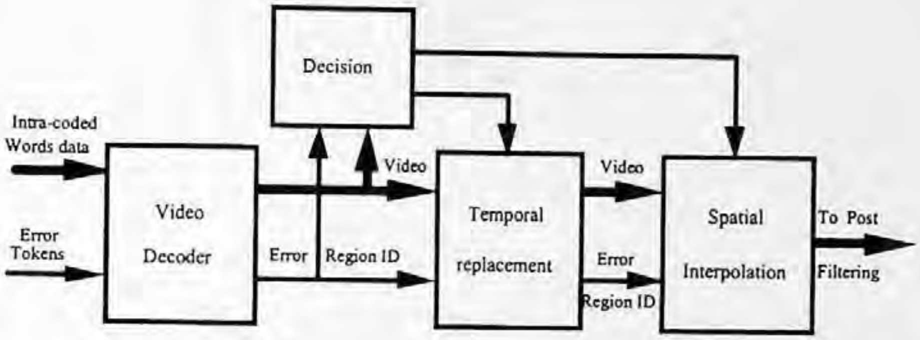


FIGURE 17.17 Two-stage error concealment strategy. (From Sun, H. and Kwok, W., *IEEE Trans. Image Proc.*, 4(4), 470–477, 1995. With permission.)

from the neighboring top and bottom macroblocks. The corrupted macroblocks are first classified into two classes according to the local activities. If local motion is smaller than spatial detail, the corrupted macroblocks are defined as the first class and will be concealed by temporal replacement; when local motion is greater than local spatial detail, the corrupted macroblocks are defined as the second class and will be concealed by spatial interpolation. The overall concealment procedure consists of two stages. First, temporal replacement is applied to all corrupted macroblocks of the first class throughout the whole frame. After the temporal replacement stage, the remaining unconcealed damaged macroblocks of the second class are more likely to be surrounded by valid image macroblocks. A stage of spatial interpolation is then performed on them. This will now result in less blurring, or the blurring will be limited to smaller areas. Therefore, a good compromise between shearing (discontinuity or shift of edge or line) and blurring can be obtained.

17.5.3 Algorithm Enhancements

As discussed above, I-picture errors, which are imperfectly concealed, will tend to propagate through all frames in the group of pictures (GOP). Therefore, it is desirable to develop enhancements for the basic spatiotemporal error concealment technique to improve further the accuracy with which missing I-picture pixels are replaced. Three new algorithms have been developed for this purpose. The first is an extension of the spatial restoration technique outlined earlier, and is based on processing of edge information in a large local neighborhood to obtain better restoration of the missing data. The second and third are variations which involve encoder modifications aimed at improved error concealment performance. Specifically, information such as I-picture pseudo-motion vectors, or low-resolution data in a hierarchical compression system are added in the encoder. These redundancies can significantly benefit error concealment in the decoders that must operate under higher cell loss/error conditions, while having a relatively modest impact on nominal image quality.

17.5.3.1 Directional Interpolation

Improvements in spatial interpolation algorithms (for use with MPEG I-pictures) have been proposed (Kwok and Sun, 1993; Sun and Kwok, 1995). In these studies, additional smoothness criteria and/or directional filtering are used for estimating the picture area to be replaced. The new algorithms utilize spatially correlated edge information from a large local neighborhood of surrounding pixels and perform directional or multidirectional interpolation to restore the missing block. The block diagram illustrating the general principle of the restoration process is shown in Figure 17.18.

Three parts are included in the restoration processing: edge classification, spatial interpolation, and pattern mixing. The function of the classifier is to select the top one, two, or three directions that strongly characterize edge orientations in the surrounding neighborhood. Spatial interpolation

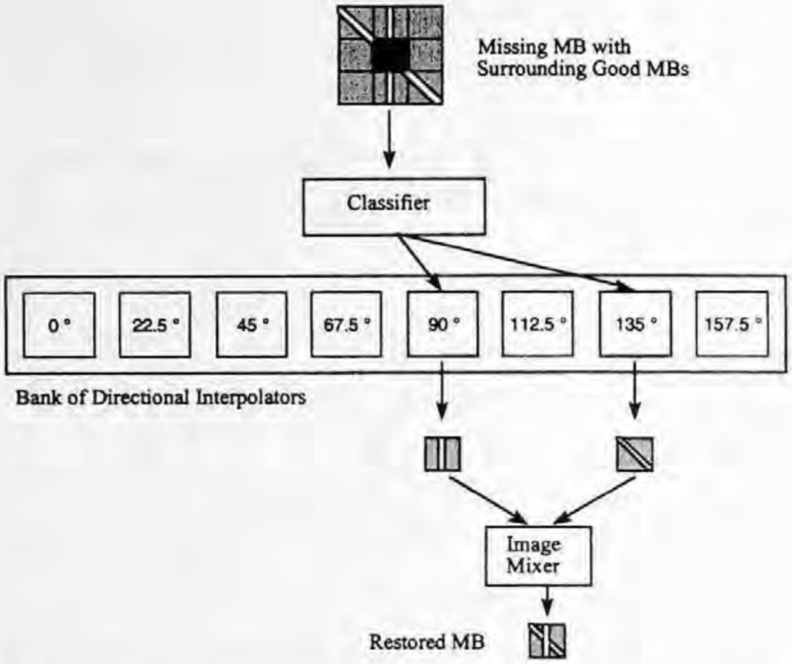


FIGURE 17.18 The multidirectional edge restoration process. (From Sun, H. and Kwok, W., *IEEE Trans. Image Proc.*, 4(4), 470–477, 1995. With permission.)

is performed for each of the directions determined by the classifier. For a given direction, a series of 1-D interpolations are carried out along that direction. All of the missing pixels are interpolated from a weighted average of good neighborhood pixels. The weights depend inversely on the distance from the missing pixel to the good neighborhood pixels. The purpose of pattern mixing is to extract strong characteristic features of two or more images and merge them into one image, which is then used to replace the corrupted one. Results show that these algorithms are capable of providing subjectively better edge restoration in missing areas, and may thus be useful for I-picture processing in high-error-rate scenarios. However, the computational practicality of these edge-filtering techniques needs further investigation for given application scenarios.

17.5.3.2 I-Picture Motion Vectors

Motion information is very useful in concealing losses in P- and B-frames, but is not available for I-pictures. This limits the concealment algorithm to spatial or direct temporal replacement options described above, which may not always be successful in moving areas of the picture. If motion vectors are made available for all MPEG frames (including intracoded ones) as an aid for error concealment (Sun et al., 1992a), good error concealment performance can be obtained without the complexity of adaptive spatial processing. Therefore, a syntax extension has been adopted by the MPEG-2 where motion vectors can be transmitted in an I-picture as the redundancy for error-concealment purposes (Sun et al., 1992b). The macroblock syntax is unchanged, however, motion vectors are interpreted in the following way: the decoded forward motion vectors belong to the macroblock spatially below the current macroblock, and describe how that macroblock can be replaced from the previous anchor frame in the event that the macroblock cannot be recovered. Simulation results have shown that subjective picture quality with I-picture motion vectors is noticeably superior to conventional temporal replacement, and that the overhead for transmitting the additional motion vectors is less than 0.7% of the total bit rate at a bit rate of about 6 to 7 Mbps.

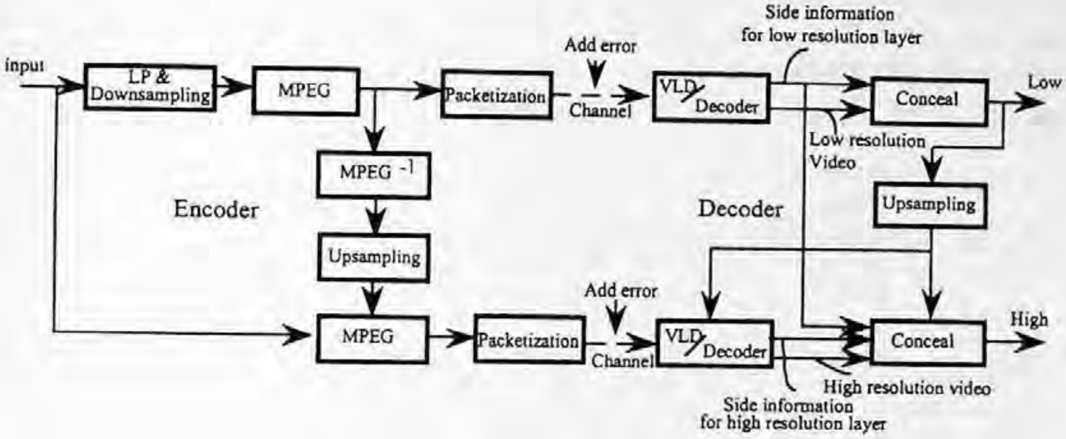


FIGURE 17.19 Block diagram of spatial scalability with error concealment.

17.5.3.3 Spatial Scalable Error Concealment

This approach for error concealment of MPEG video is based on the scalability (or hierarchy) feature of MPEG-2 (ISO/IEC, 1995). Hierarchical transmission provides more possibilities for error concealment, when a corresponding two-tier transmission media is available. A block diagram illustrating the general principle of coding system with spatial scalability and error concealment is shown in Figure 17.19.

It should be noted that the concept of scalable error concealment is different from the two-tier concept with data partitioning. Scalable concealment uses the spatial scalability feature in MPEG-2, while the two-tier case uses the data partitioning feature of MPEG-2, in which the data corresponds to the same spatial resolution layer but is partitioned to two parts with a breakpoint. In spatial scalability, the encoder produces two separate bitstreams: one for the low-resolution base layer and another for the high-resolution enhancement. The high-resolution layer is encoded with an adaptive choice of temporal prediction from previous anchor frames and compatible spatial prediction (obtained from the up-sampled low-resolution layer) corresponding to the current temporal reference. In the decoder, redundancies that exist in the scaling data greatly benefit the error concealment processing. In a simple experiment with spatially scalable MPEG-2, we consider a scenario in which losses in the high-resolution MPEG-2 video are concealed with information from the low-resolution layer. Actually, there are two kinds of information in the lower layer that can be used to conceal the data loss in the high-resolution layer: up-sampled picture data and scaled motion information. Therefore, three error concealment approaches are possible:

1. *Up-sampled substitution*: Lost data are replaced by colocated up-sampled data in the low-resolution decoded frame. The up-sampled picture is obtained from the low-resolution picture with proper up-sampling filter.
2. *Mixed substitution*: Lost macroblocks in I-picture are replaced by colocated up-sampled macroblocks in the low-resolution decoded frame, while lost macroblocks in P- and B-picture are temporally replaced by the previously decoded anchor frame with the motion vectors for the low-resolution layer.
3. *Motion vector substitution*: The previously decoded anchor frame with the motion vectors replaces lost macroblocks for the low-resolution layer appropriately scaled.

Since motion vectors are not available for I-pictures, obviously, method 3 does not work for I-pictures (unless I-picture motion vectors, concealment motion vectors, of MPEG-2 are generated

TABLE 17.4
Subjective Quality Comparison

Picture Material	Items	Alg 1	Alg 2	Alg 3	Comments
Still	Blurring	High	None	Low	Temporal replacement works very well in no-motion areas
	Shearing	None	None	None	
	Artifact blocking	Medium	None	Low	
Slow motion	Blurring	High	None	Low	Temporal replacement works well in slow-motion areas
	Shearing	None	Low	Low	
	Artifact blocking	Medium	None	Low	
Fast motion	Blurring	High	None	Medium	Temporal replacement causes more shearing; spatial interpolation results in blurring; adaptive strategy limits blurring in smaller areas
	Shearing	None	High	Low	
	Artifact blocking	High	Low	Medium	
Overall	The adaptive strategy of steering the temporal replacement and spatial interpolation according to the measures of local activity and local motion gives a good compromise between shearing and blurring				

in encoder). Simulation results have shown that, on average, the up-sampled substitution outperforms the other two, and mixed substitution also provides acceptable results in the case of video with smooth motion.

17.5.4 Summary of Error Concealment

In this section, a general class of error-concealment algorithms for MPEG video has been discussed. The error-concealment approaches that have been described are practical for current MPEG decoder implementations, and have been demonstrated to provide significant robustness. Specifically, it has been shown that the adaptive spatiotemporal algorithm can provide reasonable picture quality at cell loss ratios (CLR) as high as 10^{-3} when used in conjunction. These results confirm that compressed video is far less fragile than originally believed when appropriate transport and concealment techniques are employed. The results can be summarized as in Table 17.4.

Several concealment algorithm extensions based on directional filtering, I-picture pseudo-motion vectors, and MPEG-2 scalability were also considered and shown to provide performance gains that may be useful in certain application scenarios. In view of the practical benefits of robust video delivery, it is recommended that such error resilience functions (along with associated transport structures) be important for implementation in emerging TV, HDTV, teleconferencing, and multimedia systems if the cell loss rates on these transmission systems are significant. Particularly for terrestrial broadcasting and ATM network scenarios, we believe that robust video delivery based on decoder error concealment is an essential element of a viable system design.

17.6 SUMMARY

In this chapter, several application issues of MPEG-2 are discussed. The most successful application of MPEG-2 is the U.S. HDTV standard. The other application issues include transcoding with bitstream scaling, down-conversion decoding, and error concealment. Transcoding is a very interesting topic that converts the bitstreams between different standards. The error concealment is very useful in the noisy communication channels such as terrestrial television broadcasting. The down-conversion decoder responds to the market requirement during the DTV transition-period and long-term need for displaying DTV signals on computer monitors.

17.7 EXERCISES

17-1. In DTV applications, describe the advantages and disadvantages of interlaced format and progressive format. Explain why the computer industry favors progressive format and TV manufacturers like interlaced format.

17-2. Do all DTV formats have square pixel format? Why is square pixel format important for digital television?

17-3. The bitstream scaling is one kind of transcoding; according to your knowledge, describe several other kinds of transcoding (such as MPEG-1 to JPEG) and propose a feasible solution to achieve the transcoding requirements.

17-4. What type of MPEG-2 frames will cause a higher degree of error propagation if errors occur? What technique of error concealment is allowed by the MPEG-2 syntax? Using this technique, perform simulations with several images to determine the penalty in the case of no errors.

17-5. To reduce the drift in a down-conversion decoder, what coding parameters can be chosen at the encoder? Will these actions affect the coding performance?

17-6. What are the advantages and disadvantages of a down-conversion decoder in the frequency domain and spatial domain?

REFERENCES

Bao, J., H. Sun, and T. Poon, HDTV down-conversion decoder, *IEEE Trans. Consumer Elec.*, 42(3), 402-410, 1996.

Boyce, J., J. Henderson, and L. Pearlestien, An SDTV decoder with HDTV capability: an all-format ATV decoder, presented at SMPTE Fall Conference, New Orleans, 1995.

Bruni, R., A. Chimienti, M. Lucenteforte, D. Pau, and R. Sannino, A novel adaptive vector quantization method for memory reduction in MPEG-2 HDTV receivers, *IEEE Trans. Consumer Elec.*, 44(3), 537-544, 1998.

de With, P. H. N., P. H. Frencken, and M. v.d. Schaar-Mitrea, An MPEG decoder with embedded compression for memory reduction, *IEEE Trans. Consumer Elec.*, 44(3), 545-555, 1998.

Ghanbari, M. Two-layer coding of video signals for VBR networks, *IEEE J. Selected Areas Comm.*, 7(5), 771-781, 1989.

Gonzalez, R. C. and P. Wintz, *Digital Image Processing*, 2nd ed., Addison-Wesley, Reading, MA, 1987, 232-233.

Grand Alliance, HDTV System Specification Version 2.0, December 7, 1994.

Harthanck, W., W. Keesen, and D. Westerkamp, Concealment techniques for block encoded TV-signals, presented at Picture Coding Symposium, 1986.

Isnardi, M. A. Consumers seek easy-to-use products, *IEEE Spect.*, 64, Jan. 1993.

ISO/IEC, MPEG Test Model 5, ISO/IEC JTC/SC29/WG11 Document. April, 1993.

ISO/IEC, MPEG-2 International Standard. Video Recommendation ITU-T H.262, ISO/IEC 13818-2, Jan. 10, 1995.

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

Jeng, F.-C. and S. H. Lee, Concealment of bit error and cell loss in inter-frame coded video transmission, *ICC Proceeding*, ICC'91, 496-500, 1991.

Joseph, K., S. Ng, D. Raychaudhuri, R. Saint Girons, T. Savatier, R. Siracusa, and J. Zdepski, MPEG++: A robust compression and transport system for digital HDTV, *Signal Proc. Image Comm.*, 4, 307-323, 1992a.

Joseph, K., S. Ng, D. Raychaudhuri, R. Saint Girons, R. Siracusa, and J. Zdepski, Prioritization and transport in the ADTV digital simulcast system, *Proceedings ICCE '92*, 1992b.

Karlsson, G. and M. Vetterli, Packet video and its integration into the network architecture, *IEEE J. Selected Areas Communication*, 739-751, 1989.

Kishino, F., K. Manabe, Y. Hayashi, and H. Yasuda, Variable bit-rate coding of video signals for ATM networks, *IEEE J. Selected Areas Comm.*, 7(5), 801-806, 1989.

Kwok, W. and H. Sun, Multi-directional interpolation for spatial error concealment, *IEEE Trans. Consumer Elec.*, 455-460, 1993.

Lancaster, P. and M. Tismenetsky, *The Theory of Matrices with Application*, Academic Press, Boston, 1985

Lei, S., A quadtree embedded compression algorithm for memory saving DTV decoders, *Proceedings International Conference on Consumer Electronics*, Los Angeles, CA, June 1999.

Merhav, N. and V. Bhaskaran, Fast algorithms for DCT-domain image down-sampling and for inverse motion compensation, *IEEE Trans. Circ. Syst. Video Technol.*, 7(3), 468-476, 1997.

Mokry, R. and D. Anastassiou, Minimal error drift in frequency scalability for motion-compensated DCT coding, *IEEE Trans. Circ. Syst. Video Technol.*, 4(4), 392-406, 1994.

Ng, S. Thompson Consumer Electronics, Low Resolution HDTV Receivers, U.S. patent 5,262,854, Nov. 16, 1993.

Pang, K. K., H. G. Lim, S. Dunstan, and J. M. Badcock, Frequency domain decimation and interpolation techniques, presented at Picture Coding Symposium, Melbourne, Australia, March 1996.

Perkins, M. and D. Arnstein. Statistical multiplexing of multiple MPEG-2 video programs in a single channel, *SMPTE J.*, September 1995.

Reitmeier, G. A. The U.S. advanced television standard and its impact on VLSI, *VLSI and Signal Processing, Systems for Signal, Image, and Video Technology*, Kluwer Academic Press, 1997.

Siracusa, R., K. Joseph, J. Zdepski, and D. Raychaudhuri, Flexible and robust packet transport for digital HDTV, *IEEE J. Selected Areas Comm.*, 11(1), ISACEM, 88-98, 1993.

Sun, H. Hierarchical decoder for MPEG compressed video data *IEEE Trans. Consumer Elec.*, 39(3), 559-562, 1993.

Sun, H., K. Challapali, and J. Zdepski, Error concealment in simulcast AD-HDTV decoder, *IEEE Trans. Consumer Elec.*, 38(3), 108-118, 1992.

Sun, H., M. Uz, J. Zdepski, and R. Saint Girons, A Proposal for Increased Error Resilience, ISO-IEC/JTC1?SC29/WG11, MPEG92, Sept. 30, 1992b.

Sun, H. and W. Kwok, Restoration of damaged block transform coded image using projection onto convex sets, *IEEE Trans. Image Process.*, 4(4), IIPRE4, 470-477, 1995.

Vetro, A. and H. Sun, On the motion compensation within a down-conversion decoder, *J. Elec. Imag.*, 7(3), 1998a.

Vetro, A. and H. Sun, Frequency domain down-conversion using an optimal motion compensation scheme, *J. Imag. Sci. Technol.*, 9(4), 1998b.

Vetro, A., H. Sun, P. DaGraca, and T. Poon, Minimum drift architectures for three-layer scalable DTV decoding, *IEEE Trans. Consumer Elec.*, 44(3), 1998.

Wang, Y. and Q.-F. Zhu, Signal loss recovery in DCT-based image and video codecs, *Proceedings of SPIE on Visual Communication and Image Processing*, Boston, 667-678, Nov. 1991.

Yu, H., W.-M. Lam, B. Canfield, and B. Beyers, Block-based image processor for memory efficient MPEG video decoding, *Proceedings International Conference on Consumer Electronics*, Los Angeles, CA, June 1999.

Zdepski, J. et al., Packet transport of rate-free interframe DCT compressed digital video on a CSMA/CD LAN, *Proceedings IEEE Global Conference on Communications*, Dallas TX, Nov. 1989.

Zdepski, J. et al., Prioritized Packet Transport of VBR CCITT H.261 Format Compressed Video on a CSMA/CD LAN, Third International Workshop on Packet Video, Morristown, NJ, March 22-23, 1990.

18 MPEG-4 Video Standard: Content-Based Video Coding

This chapter provides an overview of the ISO MPEG-4 standard. The MPEG-4 work includes natural video, synthetic video, audio and systems. Both natural and synthetic video have been combined into a single part of the standard, which is referred to as MPEG-4 visual (ISO/IEC, 1998a). It should be emphasized that neither MPEG-1 nor MPEG-2 considers synthetic video (or computer graphics) and the MPEG-4 is also the first standard to consider the problem of content-based coding. Here, we focus on the video parts of the MPEG-4 standard.

18.1 INTRODUCTION

As we discussed in the previous chapters, MPEG has completed two standards: MPEG-1 that was mainly targeted for CD-ROM applications up to 1.5 Mbps and MPEG-2 for digital TV and HDTV applications at bit rates between 2 and 30 Mbps. In July 1993, MPEG started its new project, MPEG-4, which was targeted at providing technology for multimedia applications. The first working draft (WD) was completed in November 1996, and the committee draft (CD) of version 1 was completed in November 1997. The draft international standard (DIS) of MPEG-4 was completed in November of 1998, and the international standard (IS) of MPEG-4 version 1 was completed in February of 1999. The goal of the MPEG-4 standard is to provide the core technology that allows efficient content-based storage, transmission, and manipulation of video, graphics, audio, and other data within a multimedia environment. As we mentioned before, there exist several video-coding standards such as MPEG-1/2, H.261, and H.263. Why do we need a new standard for multimedia applications? In other words, are there any new attractive features of MPEG-4 that the current standards do not have or cannot provide? The answer is yes. The MPEG-4 has many interesting features that will be described later in this chapter. Some of these features are focused on improving coding efficiency; some are used to provide robustness of transmission and interactivity with the end user. However, among these features the most important one is the content-based coding. MPEG-4 is the first standard that supports content-based coding of audio visual objects. For content providers or authors, the MPEG-4 standard can provide greater reusability, flexibility, and manageability of the content that is produced. For network providers, MPEG-4 will offer transparent information, which can be interpreted and translated into the appropriate native signaling messages of each network. This can be accomplished with the help of relevant standards bodies that have the jurisdiction. For end users, MPEG-4 can provide much functionality to make the user terminal have more capabilities of interaction with the content. To reach these goals, MPEG-4 has the following important features:

The contents such as audio, video, or data are represented in the form of primitive audio visual objects (AVOs). These AVOs can be natural scenes or sounds, which are recorded by video camera or synthetically generated by computers.

The AVOs can be composed together to create compound AVOs or scenes.

The data associated with AVOs can be multiplexed and synchronized so that they can be transported through network channels with certain quality requirements.

18.2 MPEG-4 REQUIREMENTS AND FUNCTIONALITIES

Since the MPEG-4 standard is mainly targeted at multimedia applications, there are many requirements to ensure that several important features and functionalities are offered. These features include the allowance of interactivity, high compression, universal accessibility, and portability of audio and video content. From the MPEG-4 video requirement document, the main functionalities can be summarized by the following three aspects: content-based interactivity, content-based efficient compression, and universal access.

18.2.1 Content-Based Interactivity

In addition to provisions for efficient coding of conventional video sequences, MPEG-4 video has the following features of content-based interactivity.

18.2.1.1 Content-Based Manipulation and Bitstream Editing

The MPEG-4 supports the content-based manipulation and bitstream coding without the need for transcoding. In MPEG-1 and MPEG-2, there is no syntax and no semantics for supporting true manipulation and editing in the compressed domain. MPEG-4 provides the syntax and techniques to support content-based manipulation and bitstream editing. The level of access, editing, and manipulation can be done at the object level in connection with the features of content-based scalability.

18.2.1.2 Synthetic and Natural Hybrid Coding (SNHC)

The MPEG-4 supports combining synthetic scenes or objects with natural scenes or objects. This is for "compositing" synthetic data with ordinary video, allowing for interactivity. The related techniques in MPEG-4 for supporting this feature include sprite coding, efficient coding of 2-D and 3-D surfaces, and wavelet coding for still textures.

18.2.1.3 Improved Temporal Random Access

The MPEG-4 provides and efficient method to access randomly, within a limited time, and with the fine resolution parts, e.g., video frames or arbitrarily shaped image objects from an audiovisual sequence. This includes conventional random access at very low bit rate. This feature is also important for content-based bitstream manipulation and editing.

18.2.2 Content-Based Efficient Compression

One initial goal of MPEG-4 is to provide a highly efficient coding tool with high compression at very low bit rates. But this goal has now extended to a large range of bit rates from 10 Kbps to 5 Mbps, which covers QSIF to CCIR601 video formats. Two important items are included in this requirement.

18.2.2.1 Improved Coding Efficiency

The MPEG-4 video standard provides subjectively better visual quality at comparable bit rates compared with the existing or emerging standards, including MPEG-1/2 and H.263. MPEG-4 video contains many new tools, which optimize the code in different bit rate ranges. Some experimental results have shown that it outperforms MPEG-2 and H.263 at the low bit rates. Also, the content-based coding reaches the similar performance of the frame-based coding.

18.2.2.2 Coding of Multiple Concurrent Data Streams

The MPEG-4 provides the capability of coding multiple views of a scene efficiently. For stereoscopic video applications, MPEG-4 allows the ability to exploit redundancy in multiple viewing points of the same scene, permitting joint coding solutions that allow compatibility with normal video as well as the ones without compatibility constraints.

18.2.3 Universal Access

The another important feature of the MPEG-4 video is the feature of universal access.

18.2.3.1 Robustness in Error-Prone Environments

The MPEG-4 video provides strong error robustness capabilities to allow access to applications over a variety of wireless and wired networks and storage media. Sufficient error robustness is provided for low-bit-rate applications under severe error conditions (e.g., long error bursts).

18.2.3.2 Content-Based Scalability

The MPEG-4 video provides the ability to achieve scalability with fine granularity in content, quality (e.g., spatial and temporal resolution), and complexity. These scalabilities are especially intended to result in content-based scaling of visual information.

18.2.4 Summary of MPEG-4 Features

From above description of MPEG-4 features, it is obvious that the most important application of MPEG-4 will be in a multimedia environment. The media that can use the coding tools of MPEG-4 include computer networks, wireless communication networks, and the Internet. Although it can also be used for satellite, terrestrial broadcasting, and cable TV, these are still the territories of MPEG-2 video since MPEG-2 already has made such a large impact in the market. A large number of silicon solutions exist and its technology is more mature compared with the current MPEG-4 standard. From the viewpoint of coding theory, we can say there is no significant breakthrough in MPEG-4 video compared with MPEG-2 video. Therefore, we cannot expect to have a significant improvement of coding efficiency when using MPEG-4 video over MPEG-2. Even though MPEG-4 optimizes its performance in a certain range of bit rates, its major strength is that it provides more functionality than MPEG-2. Recently, MPEG-4 added the necessary tools to support interlaced material. With this addition, MPEG-4 video does support all functionalities already provided by MPEG-1 and MPEG-2, including the provision to compress efficiently standard rectangular-sized video at different levels of input formats, frame rates, and bit rates.

Overall, the incorporation of an object- or content-based coding structure is the feature that allows MPEG-4 to provide more functionality. It enables MPEG-4 to provide the most elementary mechanism for interactivity and manipulation with objects of images or video in the compressed domain without the need for further segmentation or transcoding at the receiver, since the receiver can receive separate bitstreams for different objects contained in the video. To achieve content-based coding, the MPEG-4 uses the concept of a video object plane (VOP). It is assumed that each frame of an input video is first segmented into a set of arbitrarily shaped regions or VOPs. Each such region could cover a particular image or video object in the scene. Therefore, the input to the MPEG-4 encoder can be a VOP, and the shape and the location of the VOP can vary from frame to frame. A sequence of VOPs is referred to as a video object (VO). The different VOs may be encoded into separate bitstreams. MPEG-4 specifies demultiplexing and composition syntax which provide the tools for the receiver to decode the separate VO bitstreams and composite them into a

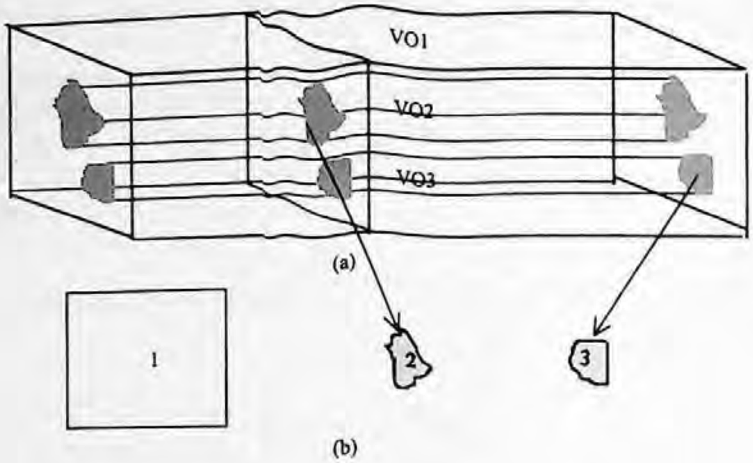


FIGURE 18.1 Video object definition and format: (a) video object, (b) VOPs.

frame. In this way, the decoders have more flexibility to edit or rearrange the decoded video objects. The detailed technical issues will be addressed in the following sections.

18.3 TECHNICAL DESCRIPTION OF MPEG-4 VIDEO

18.3.1 Overview of MPEG-4 Video

The major feature of MPEG-4 is to provide the technology for object-based compression, which is capable of separately encoding and decoding video objects. To explain the idea of object-based coding clearly, we should review the set of video object-related definitions. An image scene may contain several objects. In the example of Figure 18.1, the scene contains the background and two objects. The time instant of each video object is referred to as the VOP. The concept of a VO provides a number of functionalities of MPEG-4, which are either impossible or very difficult in MPEG-1 or MPEG-2 video coding. Each video object is described by the information of texture, shape, and motion vectors. The video sequence can be encoded in a way that will allow the separate decoding and reconstruction of the objects and allow the editing and manipulation of the original scene by simple operation on the compressed bitstream domain. The feature of object-based coding is also able to support functionality such as warping of synthetic or natural text, textures, image, and video overlays on reconstructed video objects.

Since MPEG-4 aims at providing coding tools for multimedia environments, these tools not only allow one to compress natural video objects efficiently, but also to compress synthetic objects, which are a subset of the larger class of computer graphics. The tools of MPEG-4 video includes the following:

- Motion estimation and compensation
- Texture coding
- Shape coding
- Sprite coding
- Interlaced video coding
- Wavelet-based texture coding
- Generalized temporal and spatial as well as hybrid scalability
- Error resilience.

The technical details of these tools will be explained in the following sections.

18.3.2 Motion Estimation and Compensation

For object-based coding, the coding task includes two parts: texture coding and shape coding. The current MPEG-4 video texture coding is still based on the combination of motion-compensated prediction and transform coding. Motion-compensated predictive coding is a well-known approach for video coding. Motion compensation is used to remove interframe redundancy, and transform coding is used to remove intraframe redundancy, as in the MPEG-2 video-coding scheme. However, there are lots of modifications and technical details in MPEG-4 for coding a very wide range of bit rates. Moreover, MPEG-4 coding has been optimized for low-bit-rate applications with a number of new tools. In other words, MPEG-4 video coding uses the most common coding technologies, such as motion compensation and transform coding, but at the same time, it modifies some traditional methods such as advanced motion compensation and also creates some new features, such as sprite coding.

The basic technique to perform motion-compensated predictive coding for coding a video sequence is motion estimation (ME). The basic ME method used in the MPEG-4 video coding is still the block-matching technique. The basic principle of block matching for motion estimation is to find the best-matched block in the previous frame for every block in the current frame. The displacement of the best-matched block relative to the current block is referred to as the motion vector (MV). Positive values for both motion vector components indicate that the best-matched block is on the bottom right of the current block. The motion-compensated prediction difference block is formed by subtracting the pixel values of the best-matched block from the current block, pixel by pixel. The difference block is then coded by a texture-coding method. In MPEG-4 video coding, the basic technique of texture coding is a discrete cosine transformation (DCT). The coded motion vector information and difference block information is contained in the compressed bitstream, which is transmitted to the decoder. The major issues in the motion estimation and compensation are the same as in the MPEG-1 and MPEG-2 which include the matching criterion, the size of search window (searching range), the size of matching block, the accuracy of motion vectors (one pixel or half-pixel), and inter/intramode decision. We are not going to repeat these topics and will focus on the new features in the MPEG-4 video coding. The feature of the advanced motion prediction is a new tool of MPEG-4 video. This feature includes two aspects: adaptive selection of 16 × 16 block or four 8 × 8 blocks to match the current 16 × 16 block and overlapped motion compensation for luminance block.

18.3.2.1 Adaptive Selection of 16 × 16 Block or Four 8 × 8 Blocks

The purpose of the adaptive selection of the matching block size is to enhance coding efficiency further. The coding performance may be improved at low bit rate since the bits for coding prediction difference could be greatly reduced at the limited extra cost for increasing motion vectors. Of course, if the cost of coding motion vectors is too high, this method will not work. The decision in the encoder should be very careful. For explaining the procedure of how to make decisions, we define $\{C(i,j),\ i,j = 0, 1,\ldots, N-1\}$ to be the pixels of the current block and $\{P(i,j),\ i,j = 0, 1, \ldots, N-1\}$ to be the pixels in the search window in the previous frame. The sum of absolute difference (SAD) is calculated as

$$SAD_N(x,y) = \begin{cases} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \left|C(i,j) - P(i,j)\right| - T & \text{if}(x,y) = (0,0) \\ \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \left|C(i,j) - P(i+x,j+y)\right| & \text{otherwise,} \end{cases} \tag{18.1}$$

where (x, y) is the pixel within the range of searching window, and T is a positive constant. The following steps then make the decision:

Step 1: To find $SAD_{16}(MV_x, MV_y)$;
Step 2: To find $SAD_8(MV1_x, MV1_y)$, $SAD_8(MV2_x, MV2_y)$, $SAD_8(MV3_x, MV3_y)$, and $SAD_8(MV4_x, MV4_y)$;
Step 3: If

$$\sum_{i=1}^{4} SAD_8\left(MV_{ix}, MV_{iy}\right) < SAD_{16}\left(MV_x, MV_y\right) - 128,$$

then choose 8 × 8 prediction; otherwise, choose 16 × 16 prediction.

If the 8 × 8 prediction is chosen, there are four motion vectors for the four 8 × 8 luminance blocks that will be transmitted. The motion vector for the two chrominance blocks is then obtained by taking an average of these four motion vectors and dividing the average value by a factor of two. Since each motion vector for the 8 × 8 luminance block has half-pixel accuracy, the motion vector for the chrominance block may have a sixteenth pixel accuracy.

18.3.2.2 Overlapped Motion Compensation

This kind of motion compensation is always used for the case of four 8 × 8 blocks. The case of one motion vector for a 16 × 16 block can be considered as having four identical 8 × 8 motion vectors, each for an 8 × 8 block. Each pixel in an 8 × 8 of the best-matched luminance block is a weighted sum of three prediction values specified in the following equation:

$$p'(i,j) = \left(H_0(i,j)\cdot q(i,j) + H_1(i,j)\cdot r(i,j) + H_2(i,j)\cdot s(i,j)\right)/8, \tag{18.2}$$

where division is with round-off. The weighting matrices are specified as:

$$H_0 = \begin{bmatrix} 4 & 5 & 5 & 5 & 5 & 5 & 5 & 4 \\ 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\ 5 & 5 & 6 & 6 & 6 & 6 & 5 & 5 \\ 5 & 5 & 6 & 6 & 6 & 6 & 5 & 5 \\ 5 & 5 & 6 & 6 & 6 & 6 & 5 & 5 \\ 5 & 5 & 6 & 6 & 6 & 6 & 6 & 6 \\ 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\ 4 & 5 & 5 & 5 & 5 & 5 & 5 & 4 \end{bmatrix}, \quad H_1 = \begin{bmatrix} 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 1 & 1 & 2 & 2 & 2 & 2 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 2 & 2 & 2 & 2 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \end{bmatrix}, \text{ and}$$

$$H_2 = \begin{bmatrix} 2 & 1 & 1 & 1 & 1 & 1 & 1 & 2 \\ 2 & 2 & 1 & 1 & 1 & 1 & 2 & 2 \\ 2 & 2 & 1 & 1 & 1 & 1 & 2 & 2 \\ 2 & 2 & 1 & 1 & 1 & 1 & 2 & 2 \\ 2 & 2 & 1 & 1 & 1 & 1 & 2 & 2 \\ 2 & 2 & 1 & 1 & 1 & 1 & 2 & 2 \\ 2 & 2 & 1 & 1 & 1 & 1 & 2 & 2 \\ 2 & 1 & 1 & 1 & 1 & 1 & 1 & 2 \end{bmatrix}$$

It is noted that $H_0(i,j) + H_1(i,j) + H_2(i,j) = 8$ for all possible (i,j). The value of $q(i,j)$, $r(i,j)$, and $s(i,j)$ are the values of the pixels in the previous frame at the locations,

$$\begin{aligned} q(i,j) &= p\left(i+MV_x^0, j+MV_y^0\right),\\ r(i,j) &= p\left(i+MV_x^1, j+MV_y^1\right), \qquad (18.3)\\ s(i,j) &= p\left(i+MV_x^2, j+MV_y^2\right), \end{aligned}$$

where (MV_x^0, MV_y^0) is the motion vector of the current 8 × 8 luminance block p(i,j), (MV_x^1, MV_y^1) is the motion vector of the block either above (for j = 0,1,2,3) or below (for j = 4,5,6,7) the current block and (MV_x^2, MV_y^2) is the motion vector of the block either to the left (for i = 0,1,2,3) or right (for i = 4,5,6,7) of the current block. The overlapped motion compensation can reduce the prediction noise at a certain level.

18.3.3 Texture Coding

Texture coding is used to code the intra-VOPs and the prediction residual data after motion compensation. The algorithm for video texture coding is based on the conventional 8 × 8 DCT with motion compensation. DCT is performed for each luminance and chrominance block, where the motion compensation is performed only on the luminance blocks. This algorithm is similar to those in H.263 and MPEG-1 as well as MPEG-2. However, MPEG-4 video texture coding has to deal with the requirement of object-based coding, which is not included in the other video-coding standards. In the following we will focus on the new features of the MPEG-4 video coding. These new features include the intra-DC and AC prediction for I-VOP and P-VOP, the algorithm of motion estimation and compensation for arbitrary shape VOP, and the strategy of arbitrary shape texture coding. The definitions of I-VOP, P-VOP, and B-VOP are similar to the I-picture, P-picture, and B-picture in Chapter 16 for MPEG-1 and MPEG-2.

18.3.3.1 Intra-DC and AC Prediction

In the intramode coding, the predictive coding is not only applied on the DC coefficients but also the AC coefficients to increase the coding efficiency. The adaptive DC prediction involves the selection of the quantized DC (QDC) value of the immediately left block or the immediately above block. The selection criterion is based on comparison of the horizontal and vertical DC gradients around the block to be coded. Figure 18.2 shows the three surrounding blocks "A," "B," and "C" to the current block "X" whose QDC is to be coded where block "A", "B," and "C" are the immediately left, immediately left and above, and immediately above block to the "X," respectively. The QDC value of block "X," QDC$_X$, is predicted by either the QDC value of block "A," QDC$_A$,

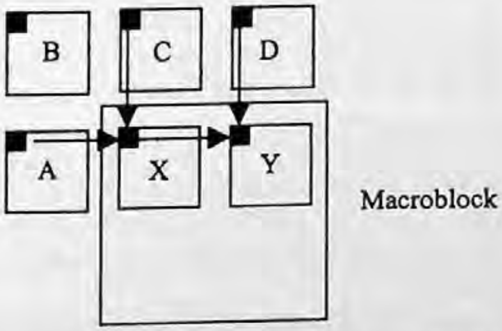


FIGURE 18.2 Previous neighboring blocks used in DC prediction. (From ISO/IEC 14496-2 Video Verification Model V.12, N2552, Dec. 1998. With permission.)

or the QDC value of block "C," QDC_C, based on the comparison of horizontal and vertical gradients as follows:

$$\begin{array}{lll} \text{If} & |QDC_A - QDC_B| < |QDC_B - QDC_C|, & QDC_P = QDC_C; \\ \text{Otherwise} & QDC_P = QDC_A. & \end{array} \tag{18.4}$$

The differential DC is then obtained by subtracting the DC prediction, QDC_P, from QDC_X. If any of block "A", "B," or "C" are outside of the VOP boundary, or they do not belong to an intracoded block, their QDC value are assumed to take a value of 128 (if the pixel is quantized to 8 bits) for computing the prediction. The DC prediction is performed similarly for the luminance and each or the two chrominance blocks.

For AC coefficient prediction, either coefficients from the first row or the first column of a previous coded block are used to predict the cosited (same position in the block) coefficients in the current block. On a block basis, the same rule for selecting the best predictive direction (vertical or horizontal direction) for DC coefficients is also used for the AC coefficient prediction. A difference between DC prediction and AC prediction is the issue of quantization scale. All DC values are quantized to the 8 bits for all blocks. However, the AC coefficients may be quantized by the different quantization scales for the different blocks. To compensate for differences in the quantization of the blocks used for prediction, scaling of prediction coefficients becomes necessary. The prediction is scaled by the ratio of the current quantization step size and the quantization step size of the block used for prediction. In the cases when AC coefficient prediction results in a larger range of prediction errors as compared with the original signal, it is desirable to disable the AC prediction. The decision of AC prediction switched on or off is performed on a macroblock basis instead of a block basis to avoid excessive overhead. The decision for switching on or off AC prediction is based on a comparison of the sum of the absolute values of all AC coefficients to be predicted in a macroblock and that of their predicted differences. It should be noted that the same DC and AC prediction algorithm is used for the intrablocks in the intercoded VOP. If any blocks used for prediction are not intrablocks, the QDC and QAC values used for prediction are set to 128 and 0 for DC and AC prediction, respectively.

18.3.3.2 Motion Estimation/Compensation of Arbitrarily Shaped VOP

In previous sections we discussed the general issues of motion estimation (ME) and motion compensation (MC). Here we are going to discuss the ME and MC for coding the texture in the arbitrarily shaped VOP. In an arbitrarily shaped VOP, the shape information is given by either binary shape information or alpha components of a gray-level shape information. If the shape information is available to both encoder and decoder, three important modifications have to be considered for the arbitrarily shaped VOP. The first is for the blocks, which are located in the border of VOP. For these boundary blocks, the block-matching criterion should be modified. Second, a special padding technique is required for the reference VOP. Finally, since the VOPs have arbitrary shapes rather than rectangular shapes, and the shapes change from time to time, an agreement on a coordinate system is necessary to ensure the consistency of motion compensation. At the MPEG-4 video, the absolute frame coordinate system is used for referencing all of the VOPs. At each particular time instance, a bounding rectangle that includes the shape of that VOP is defined. The position of upper-left corner in the absolute coordinate in the VOP spatial reference is transmitted to the decoder. Thus, the motion vector for a particular block inside a VOP is referred to as the displacement of the block in absolute coordinates.

Actually, the first and second modifications are related since the padding of boundary blocks will affect the matching of motion estimation. The purpose of padding aims at more accurate block matching. In the current algorithm, the repetitive padding is applied to the reference VOP for performing motion estimation and compensation. The repetitive padding process is performed as the following steps:

Define any pixel outside the object boundary as a zero pixel.

Scan each horizontal line of a block (one 16 × 16 for luminance and two 8 × 8 for chrominance). Each scan line is possibly composed of two kinds of line segments: zero segments and nonzero segment. It is obvious that our task is to pad zero segments. There are two kinds of zero segments: (1) between an end point of the scan line and the end point of a nonzero segment, and (2) between the end points of two different nonzero segments. In the first case, all zero pixels are replaced by the pixel value of the end pixel of nonzero segment; for the second kind of zero segment, all zero pixels take the averaged value of the two end pixels of the nonzero segments.

Scan each vertical line of the block and perform the identical procedure as described for the horizontal line.

If a zero pixel is located at the intersection of horizontal and vertical scan lines, this zero pixel takes the average of two possible values.

For the rest of zero pixels, find the closest nonzero pixel on the same horizontal scan line and the same vertical scan line (if there is a tie, the nonzero pixel on the left or the top of the current pixel is selected). Replace the zero pixel by the average of these two nonzero pixels.

For a fast-moving VOP, padding is further extended to the blocks outside the VOP but immediately next to the boundary blocks. These blocks are padded by replacing the pixel values of adjacent boundary blocks. This extended padding is performed in both horizontal and vertical directions. Since block matching is replaced by polygon matching for the boundary blocks of the current VOP, the SAD values are calculated by the modified formula:

$$SAD_N(x,y)=\begin{cases}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left|c(i,j)-p(i,j)\right|\cdot\alpha(i,j)-C & \text{if}\quad (x,y)=(0,0);\\ \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left|c(i,j)-p(i+x,j+y)\right|\cdot\alpha(i,j)-C & \text{otherwise,}\end{cases}\qquad(18.5)$$

where $C = N_B/2 + 1$ and N_B is the number of pixels inside the VOP and in this block and $\alpha(i,j)$ is the alpha component specifying the shape information, and it is not equal to zero here.

18.3.3.3 Texture Coding of Arbitrarily Shaped VOP

During encoding the VOP is represented by a bounding rectangle that is formed to contain the video object completely but with minimum number of macroblocks in it, as shown in Figure 18.3. The detailed procedure of VOP rectangle formation is given in MPEG-4 video VM (ISO/IEC, 1998b).

There are three types of macroblocks in the VOP with arbitrary shape: the macroblocks that are completely located inside of the VOP, the macroblocks that are located along the boundary of the VOP, and the macroblocks outside of the boundary. For the first kind of macroblock, there is no need for any particular modified technique to code them and just use of normal DCT with entropy coding of quantized DCT coefficients such as coding algorithm in H.263 is sufficient. The second kind of macroblocks, which are located along the boundary, contains two kinds of 8 × 8 blocks: the blocks lie along the boundary of VOP and the blocks do not belong to the arbitrary shape but lie inside the rectangular bounding box of the VOP. The second kind of blocks are referred

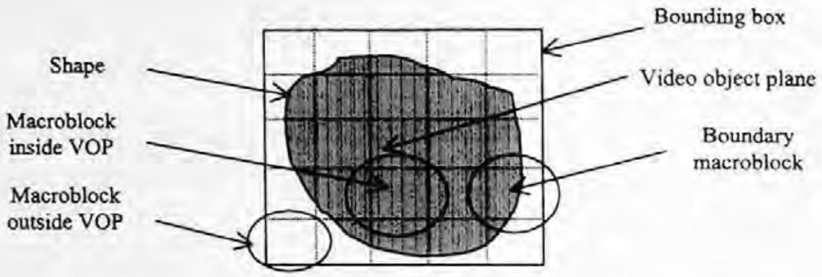


FIGURE 18.3 A VOP is represented by a bounding rectangular box.

to as transparent blocks. For those 8 × 8 blocks that do lie along the boundary of VOP, there are two different methods that have been proposed: low-pass extrapolation (LPE) padding and shape-adaptive DCT (SA-DCT). All blocks in the macroblock outside of boundary are also referred to as transparent blocks. The transparent blocks are skipped and not coded at all.

1. Low-pass extrapolation padding technique: This block-padding technique is applied to intracoded blocks, which are not located completely within the object boundary. To perform this padding technique we first assign the mean value of those pixels that are located in the object boundary (both inside and outside) to each pixel outside the object boundary. Then an average operation is applied to each pixel $p(i,j)$ outside the object boundary starting from the upper-left corner of the block and proceeding row by row to the lower-right corner pixel:

$$p(i,j) = \left[p(i,j-1) + p(i-1,j) + p(i,j+1) + p(i+1,j)\right]/4. \tag{18.6}$$

   If one or more of the four pixels used for filtering are outside of the block, the corresponding pixels are not considered for the average operation and the factor ¼ is modified accordingly.
2. SA-DCT: The shape-adaptive DCT is only applied to those 8 × 8 blocks that are located on the object boundary of an arbitrarily shaped VOP. The idea of the SA-DCT is to apply 1-D DCT transformation vertically and horizontally according to the number of active pixels in the row and column of the block, respectively. The size of each vertical DCT is the same as the number of active pixels in each column. After vertical DCT is performed for all columns with at least one active pixel, the coefficients of the vertical DCTs with the same frequency index are lined up in a row. The DC coefficients of all vertical DCTs are lined up in the first row, the first-order vertical DCT coefficients are lined up in the second row, and so on. After that, horizontal DCT is applied to each row. As the same as for the vertical DCT, the size of each horizontal DCT is the same as the number of vertical DCT coefficients lined up in the particular row. The final coefficients of SA-DCT are concentrated into the upper-left corner of the block. This procedure is shown in the Figure 18.4.

The final number of the SA-DCT coefficients is identical to the number of active pixels of the image. Since the shape information is transmitted to the decoder, the decoder can perform the inverse shape-adapted DCT to reconstruct the pixels. The regular zigzag scan is modified so that the nonactive coefficient locations are neglected when counting the runs for the run-length coding of the SA-DCT coefficients. It is obvious that for a block with all 8 × 8 active pixels, the SA-DCT becomes a regular 8 × 8 DCT and the scanning of the coefficients is identical to the zigzag scan. All SA-DCT coefficients are quantized and coded in the same way as the regular DCT coefficients

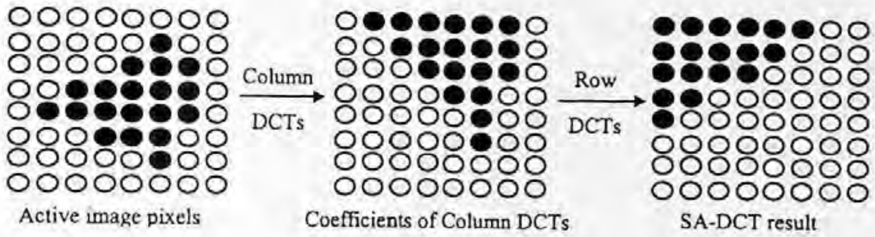


FIGURE 18.4 Illustration of SA-DCT. (From ISO/IEC 14496-2 Video Verification Model V.12, N2552, Dec. 1998. With permission.)

employing the same quantizers and VLC code tables. The SA-DCT is not included in MPEG-4 video version 1, but it is being considered for inclusion into version 2.

18.3.4 Shape Coding

Shape information of the arbitrarily shaped objects is very useful not only in the field of image analysis, computer vision, and graphics, but also in object-based video coding. MPEG-4 video coding is the first to make an effort to provide a standardized approach to compress the shape information of objects and contain the compressed results within a video bitstream. In the current MPEG-4 video coding standard, the video data can be coded on an object basis. The information in the video signal is decomposed to shape, texture, and motion. This information is then coded and transmitted within the bitstream. The shape information is provided in binary format or gray scale format. The binary format of shape information consists of a pixel map, which is generally the same size as the bounding box of the corresponding VOP. Each pixel takes on one of two possible values indicating whether it is located within the video object or not. The gray scale format is similar to the binary format with the additional feature that each pixel can take on a range of values, i.e., times an alpha value. Alpha typically has a normalized value of 0 to 1. The alpha value can be used to blend two images on a pixel-by-pixel basis in this way: new pixel = (alpha)(pixel A color) + (1 – alpha)(pixel B color).

Now let us discuss how to code the shape information. As we mentioned, the shape information is classified as binary shape or gray scale shape. Both binary and gray scale shapes are referred to as an alpha plane. The alpha plane defines the transparency of an object. Multilevel alpha maps are frequently used to blend different images. A binary alpha map defines whether or not a pixel belongs to an object. The binary alpha planes are encoded by modified content-based arithmetic encoding (CAE), while the gray scale alpha planes are encoded by motion-compensated DCT coding, which is similar to texture coding. For binary shape coding, a rectangular box enclosing the arbitrarily shaped VOP is formed as shown in Figure 18.3. The bounded rectangle box is then extended in both vertical and horizontal directions on the right-bottom side to the multiple of 16 × 16 blocks. Each 16 × 16 block within the rectangular box is referred to as binary alpha block (BAB). Each BAB is associated with colocated macroblock. The BAB can be classified as three types: transparent block, opaque block, and alpha or shape block. The transparent block does not contain any information about an object. The opaque block is entirely located inside the object. The alpha or shape block is located in the area of the object boundary; i.e., a part of block is inside of object and the rest of block is in background. The value of pixels in the transparent region is zero. For shape coding, the type information will be included in the bitstream and signaled to the decoder as a macroblock type. But only the alpha blocks need to be processed by the encoder and decoder. The methods used for each shape format contain several encoding modes. For example, the binary shape information can be encoded using either an intra- or intermode. Each of these modes can be further divided into lossy and lossless options. Gray scale shape information also contains intra- and intermodes; however, only a lossy option is used.

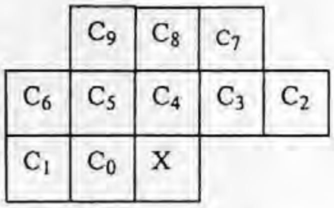


FIGURE 18.5 Template for defining the context of the pixel, X, to be coded in intramode. (From ISO/IEC 14496-2 Video Verification Model V.12, N2552, Dec. 1998. With permission.)

18.3.4.1 Binary Shape Coding with CAE Algorithm

As mentioned previously, the CAE is used to code each binary pixel of the BAB. For a P-VOP, the BAB may be encoded in intra- or intermode. Pixels are coded in scan-line order, i.e., row by row for both modes. The process for coding a given pixel includes three steps: (1) compute a context number, (2) index a probability table using the context number, and (3) use the indexed probability to drive an arithmetic encoder. In intramode, a template of 10 pixels is used to define the causal context for predicting the shape value of the current pixel as shown in Figure 18.5. For the pixels in the top and left boundary of the current macroblock, the template of causal context will contain the pixels of the already transmitted macroblocks on the top and on the left side of the current macroblock. For the two rightmost columns of the VOP, each undefined pixel such as C_7, C_3, and C_2, of the context is set to the value of its closest neighbor inside the macroblock, i.e., C_7 will take the value of C_8 and C_3 and C_2 will take the value of C_4.

A 10-bit context is calculated for each pixel, X as

$$C = \sum_{k=0}^{9} C_k \cdot 2^k . \tag{18.7}$$

This causal context is used to predict the shape value of the current pixel. For encoding the state transition, a context-based arithmetic encoder is used. The probability table of the arithmetic encoder for the 1024 contexts was derived from sequences that are outside of the test set. Two bytes are allocated to describe the symbol probability for each context; the table size is 2048 bytes. To increase coding efficiency and rate control, the algorithm allows lossy shape coding. In lossy shape coding a macroblock can be down-sampled by a factor of two or four resulting in a subblock of size 8 × 8 pixels or 4 × 4 pixels, respectively. The subblock is then encoded using the same method as for full-size block. The down-sampling factor is included in the encoded bitstream and then transmitted to the decoder. The decoder decodes the shape data and then up-samples the decoded subblock to full macroblock size according to the down-sampling factor. Obviously, it is more efficient to code shape using a high down-sampling factor, but the coding errors may occur in the decoded shape after up-sampling. However, in the case of low-bit-rate coding, lossy shape coding may be necessary since the bit budget may not be enough for lossless shape coding. Depending on the up-sampling filter, the decoded shape can look somewhat blocky. Several up-sampling filters were investigated. The best-performing filter in terms of subjective picture quality is an adaptive nonlinear up-sampling filter. It should be noted that the coding efficiency of shape coding also depends on the orientation of the shape data. Therefore, the encoder can choose to code the block as described above or transpose the macroblock prior to arithmetic coding. Of course, the transpose information has to be signaled to the decoder.

For shape coding in a P-VOP or B-VOP, the intermode may be used to exploit the temporal redundancy in the shape information with motion compensation. For motion compensation, a 2-D integer pixel motion vector is estimated using full search for each macroblock in order to minimize

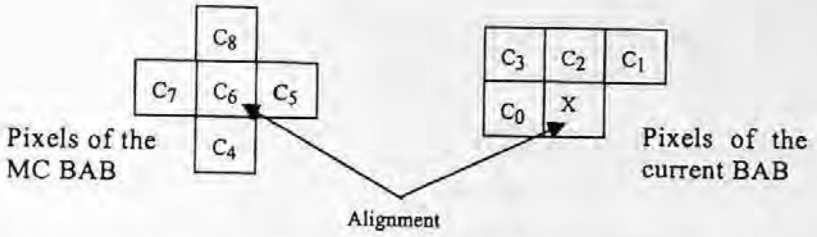


FIGURE 18.6 Template for defining the context of the pixel, X, to be coded in intermode. (From ISO/IEC 14496-2 Video Verification Model, N2552, Dec. 1998. With permission.)

the prediction error between the previously coded VOP shape and the current VOP shape. The shape motion vectors are predictively encoded with respect to the shape motion vectors of neighboring macroblocks. If no shape motion vector is available, texture motion vectors are used as predictors. The template for intermode differs from the one used for intramode. The intermode template contains 9 pixels among which 5 pixels are located in the previous frame and 4 are the current neighbors as shown in Figure 18.6.

The intermode template defines a context of 9 pixels. Accordingly, a 9-bit context or 512 contexts, can be computed in a similar way to Equation 18.7:

$$C=\sum_{k=0}^{8} C_k \cdot 2^k. \tag{18.8}$$

The probability for one symbol is also described by 2 bytes giving a probability table size of 1024 bytes. The idea of lossy coding can also be applied to the intermode shape coding by down-sampling the original BABs. For intermode shape coding, the total bits for coding the shape consist of two parts, one part for coding motion vectors and another for prediction residue. The encoder may decide that the shape representation achieved by just using motion vectors is sufficient; thus bits for coding the prediction error can be saved. Actually, there are seven modes to code the shape information of each macroblock: (1) transparent, (2) opaque, (3) intra, inter (4) with and (5) without shape motion vectors, and inter (6) with and (7) without shape motion vectors and prediction error coding. These different options with optional down-sampling and transposition allow for encoder implementations of different coding efficiency and implementation complexity. Again, this is a problem of encoder optimization, which does not belong to the standard.

18.3.4.2 Gray Scale Shape Coding

The gray scale shape information is encoded by separately encoding the shape and transparency information as shown in Figure 18.7. For a transparent object, the shape information is referred to as the support function and is encoded using the binary shape-coding method. The transparency or alpha values are treated as the texture of luminance and encoded using padding, motion compensation, and the same 8 × 8 block DCT approach for the texture coding. For an object with varying alpha maps, shape information is encoded in two steps. The boundary of the object is first losslessly encoded as a binary shape, and then the actual alpha map is encoded as texture coding.

The binary shape coding allows one to describe objects with constant transparency, while gray scale shape coding can be used to describe objects with arbitrary transparency, providing for more flexibility for image composition. One application example is a gray scale alpha shape that consists of a binary alpha shape with the value around the edges tapered from 255 to 0 to provide for a smooth composition with the background. The description of each video object layer includes the information to give instruction for selecting one of six modes for feathering. These six modes

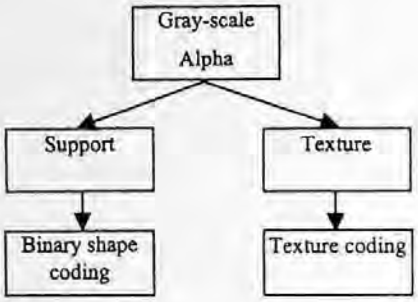


FIGURE 18.7 Gray scale shape coding.

include (1) no effects, (2) linear feathering, (3) constant alpha, (4) linear feathering and constant alpha, (5) feathering filter, and (6) feathering filter and constant alpha. The detailed description of the function of these modes are given in the reference of version 12 (ISO/IEC, 1998b).

18.3.5 Sprite Coding

As mentioned previously, MPEG-4 video has investigated a number of new tools, which attempt to improve the coding efficiency at low bit rates compared with MPEG-1/2 video coding. Among these tools, sprite coding is an efficient technology to reach this goal. A sprite is a specially composed video object that is visible throughout an entire piece of video sequence. For example, the sprite generated from a panning sequence contains all the visible pixels of the background throughout the video sequence. Portions of the background may not be seen in certain frames due to the occlusion of the foreground objects or the camera motion. This particular example is one of the static sprites. In other words, a static sprite is a possible still image. Since the sprite contains all visible background scenes of a segment video sequence where the changes within the background content are mainly caused by camera parameters, the sprite can be used for direct reconstruction of the background VOPs or as the prediction of the background VOPs within the video segment. The sprite-coding technology first efficiently transmits this background to the receiver and then stores it in a frame at both encoder and decoder. The camera parameters are then transmitted to the decoder for each frame so that the appropriate part of the background scene can be either used as the direct reconstruction or as the prediction of the background VOP. Both cases can significantly save the coding bits and increase the coding efficiency. There are two types of sprites, static sprite and dynamic sprite, which are being considered as coding tools for MPEG-4 video. A static sprite is used for a video sequence in which the objects in a scene can be separated into foreground objects and a static background. A static sprite is a special VOP, which is generated by copying the background from a video sequence. This copying includes the appropriate warping and cropping. Therefore, a static sprite is always built off-line. In contrast, a dynamic sprite is dynamically built during the predictive coding. It can be built either online or off-line. The static sprite has shown significant coding gain over existing compression technology for certain video sequences. The dynamic sprite is more complicated in the real-time application due to the difficulty of updating the sprite during the coding. Therefore, the dynamic sprite has not been adopted by version 1 of the standard. Additionally, both sprites are not easily applied to the generic scene content. Also, there is another kind of classification of sprite coding according to the method of sprite generation, namely, off-line and online sprites. Off-line is always used for static sprite generation. Off-line sprites are well suited for synthetic objects and objects that mostly undergo rigid motion. Online sprites are only used for dynamic sprites. Online sprites provide a no-latency solution in the case of natural sprite objects. Off-line dynamic sprites provide an enhanced predictive coding environment. The sprite is built with a similar way in both off-line and online methods. In particular, the same global motion estimation algorithm is exploited. The difference is that the off-line sprite is

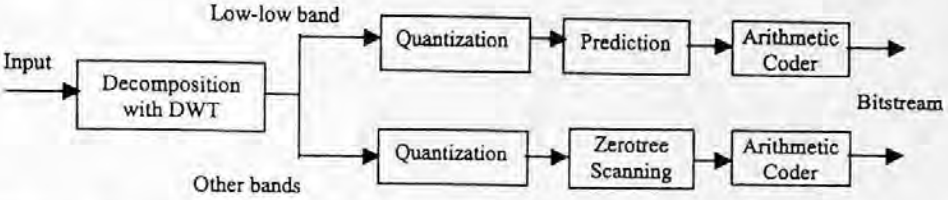


FIGURE 18.8 Block diagram of encoder of wavelet-based texture coding, DWT stands for discrete wavelet transform.

built before starting the encoding process while, in the online sprite case, both the encoder and the decoder build the same sprite from reconstructed VOPs. This is why the online dynamic sprites are more complicated in the implementation. The online sprite is not included in version 1, and will most likely not be considered for version 2 either. In sprite coding, the chrominance components are processed in the same way as the luminance components, with the properly scaled parameters according to the video format.

18.3.6 Interlaced Video Coding

Since June of 1997, MPEG-4 has extended its application to support interlaced video. Interlaced video consists of two fields per frame, which are referred to as the even field and the odd field. MPEG-2 has a number of tools, which are used to deal with field structure of video signals. These tools include frame/field-adaptive DCT coding and frame/field-adaptive motion compensation. However, the field issue in MPEG-4 has to be considered on a VOP basis instead of the conventional frame basis. When field-based motion compensation is specified, two field motion vectors and the corresponding reference fields are used to generate the prediction from each reference VOP. Shape information has to be considered in the interlaced video for MPEG-4.

18.3.7 Wavelet-Based Texture Coding

In MPEG-4 there is a texture-coding mode which is used to code the texture or still image such as in JPEG. The basic technique used in this mode is wavelet-based transform coding. The reason for adopting wavelet transform instead of DCT for still texture coding is not only its high coding efficiency, but also because the wavelet can provide excellent scalability, both spatial scalability and SNR scalability. Since the principle of wavelet-based transform coding for image compression has been explained in Chapter 8, we just briefly describe the coding procedure of this mode. The block diagram of the encoder is shown in Figure 18.8.

18.3.7.1 Decomposition of the Texture Information

The texture or still image is first decomposed into bands using a bank of analysis filters. This decomposition can be applied recursively on the obtained bands to yield a decomposition tree of subbands. An example of decomposition to depth 2 is shown in Figure 18.9.

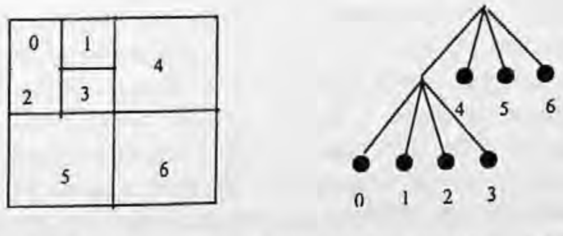
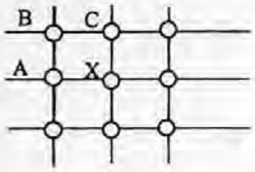


FIGURE 18.9 An example of wavelet decomposition of depth 2.



If $|W_A\text{-}|W_B| < |W_B\text{-}W_C|$, $W_{X_p} = W_C$, else $W_{X_p} = W_A$.

FIGURE 18.10 Adaptive DPCM coding of the coefficients in the lowest band.

18.3.7.2 Quantization of Wavelet Coefficients

After decomposition, the coefficients of the lowest band are coded independently of the other bands. These coefficients are quantized using a uniform midriser quantizer. The coefficients of high bands are quantized with a multilevel quantization. The multilevel quantization provides a very flexible approach to support the correct trade-off between levels and type of scalability, complexity, and coding efficiency for any application. All quantizers for the higher bands are uniform midrise quantizers with a dead zone that is twice the quantizer step size. The levels and quantization steps are determined by the encoder and specified in the bitstream. To achieve scalability, a bi-level quantization scheme is used for all multiple quantizers. This quantizer is also uniform and midrise with a dead zone that is twice the quantization step. The coefficients outside of the dead zone are quantized to 1 bit. The number of quantizers is equal to the maximum number of bit planes in the wavelet transform representation. In this bi-level quantizer, the maximum number of bit planes instead of a quantization step size is specified in the bitstream.

18.3.7.3 Coding of Wavelet Coefficients of Low–Low Band and Other Bands

The quantized coefficients at the lowest band are DPCM coded. Each of the current coefficients is predicted from three other quantized coefficients in its neighborhood in a way shown in Figure 18.10.

The coefficients in high bands are coded with the zerotree algorithm (Shapiro, 1993), which has been discussed in Chapter 8.

18.3.7.4 Adaptive Arithmetic Coder

The quantized coefficients and the symbols generated by the zerotree are coded using an adaptive arithmetic coder. In the arithmetic coder three different tables which correspond to the different statistical models have been utilized. The method used here is very similar to one in Chapter 8. Further detail can be found in MPEG-4 (ISO/IEC, 1998a).

18.3.8 Generalized Spatial and Temporal Scalability

The scalability framework is referred to as generalized scalability that includes the spatial and the temporal scalability similar to MPEG-2. The major difference is that MPEG-4 extends the concept of scalability to be content based. This unique functionality allows MPEG-4 to be able to resolve objects into different VOPs. By using the multiple VOP structure, different resolution enhancement can be applied to different portions of a video scene. Therefore, the enhancement layer may only be applied to a particular object or region of the base layer instead of to the entire base layer. This is a feature that MPEG-2 does not have.

In spatial scalability, the base layer and the enhancement layer can have different spatial resolutions. The base-layer VOPs are encoded in the same way as the nonscalable encoding technique described previously. The VOPs in the enhancement layer are encoded as P-VOPs or B-VOPs, as shown in Figure 18.11. The current VOP in the enhancement layer can be predicted

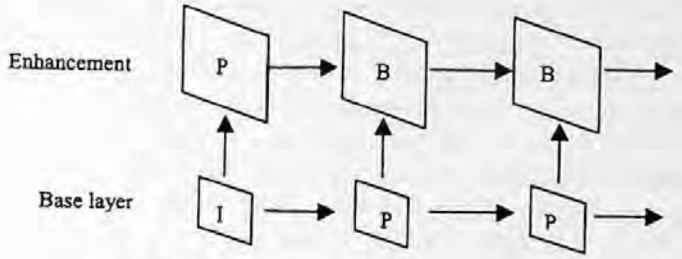


FIGURE 18.11 Illustration of spatial scalability.

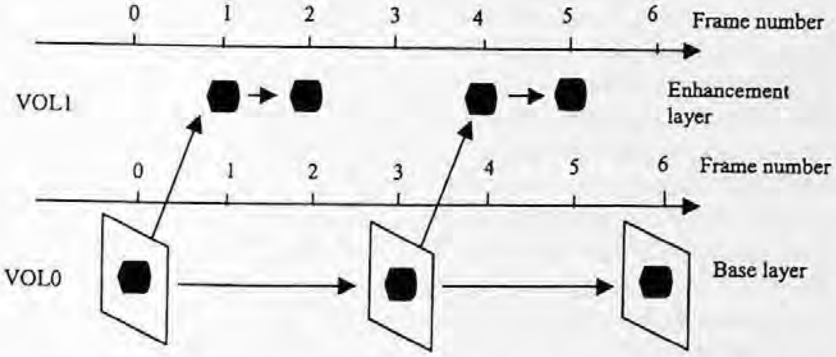


FIGURE 18.12 An example of temporal scalability. (From ISO/IEC 14496-2 Video Verification Model V.12, N2552, Dec. 1998. With permission.)

from either the up-sampled base layer VOP or the previously decoded VOP at the same layer as well as both of them. The down-sampling and up-sampling processing in spatial scalability is not a part of the standard and can be defined by the user.

In temporal scalability, a subsequence of subsampled VOP in the time domain is coded as a base layer. The remaining VOPs can be coded as enhancement layers. In this way, the frame rate of a selected object can be enhanced so that it has a smoother motion than other objects. An example of temporal scalability is illustrated in Figure 18.12. In Figure 18.12, the VOL_0 is the entire frame with both an object and a background, while VOL_1 is a particular object in VOL_0. VOL_0 is encoded with a low frame rate and VOL_1 is the enhancement layer. The high frame rate can be reached for the particular object by combining the decoded data from both the base layer and the enhancement layer. Of course, the B-VOP is also used in temporal scalability for coding the enhancement layer, which is another type of temporal scalability. As in spatial scalability, the enhancement layer can be used to improve either the entire base layer frame resolution or only a portion of the base layer resolution.

18.3.9 Error Resilience

The MPEG-4 visual coding standard provides error robustness and resilience to allow access of image and video data over a wide range of storage and transmission media. The error resilience tool development effort is divided into three major areas, which include resynchronization, data recovery, and error concealment. As with other coding standards, MPEG-4 makes heavy use of variable-length coding to reach high coding performance. However, if even 1 bit is lost or damaged, the entire bitstream becomes undecodable due to loss of synchronization. The resynchronization tools attempt to enable resynchronization between the decoder and the bitstream after a transmission error or errors have been detected. Generally, the data between the synchronization point prior to the error and the first point, where synchronization is reestablished, are discarded. The purpose of resynchronization is to localize effectively the amount of data discarded by the decoder; then the other methods such as error concealment can be used to conceal the damaged areas of a decoded picture. Currently, the resynchronization approach adopted by MPEG-4 is referred to as a packet approach. This approach is similar to the group of block (GOB) structure used in H.261 and H.263. In the GOB structure, the GOB contains a start code, which provides the location information of the GOB. MPEG-4 adopted a similar approach in which a resynchronization marker is periodically inserted into the bitstream at the particular macroblock locations. The resynchronization marker is used to indicate the start of new video packet. This marker is distinguished from all possible VLC codewords as well as the VOP start code. The packet header information is then provided at the start of a video packet. The header contains the information necessary to restart the decoding process. This includes the macroblock number of the first macroblock contained in this packet and the quantization parameter necessary to decode the first macroblock. The macroblock number provides the necessary spatial resynchronization while the quantization parameter allows the differential decoding process to be resynchronized. It should be noted that when error resilience is used within MPEG-4, some of the compression efficiency tools need to be modified. For example, all predictively encoded information must be contained within a video packet to avoid error propagation. In conjunction with the video packet approach to resynchronization, MPEG-4 has also adopted a fixed-interval synchronization method which requires that VOP start-codes and resynchronization markers appear only at legal fixed-interval locations in the bitstream. This will help to avoid the problems associated with start-code emulation. In this case, when fixed-interval synchronization is utilized, the decoder is only required to search for a VOP start-code at the beginning of each fixed interval. The fixed-interval synchronization method extends this approach to any predetermined interval.

After resynchronization is reestablished, the major problem is recovering lost data. A new tool called reversible variable-length codes (RVLC) is developed for the purpose of data recovery. In this approach, the variable-length codes are designed such that the codes can be read both in the forward and the reverse direction. An example of such a code includes codewords like 111, 101, 010. All these codewords can be read reversibly. However, it is obvious that this approach will reduce the coding efficiency that is achieved by the entropy coder. Therefore, this approach is used only in the cases where error resilience is important.

Error concealment is an important component of any error-robust video coding. The error-concealment strategy is highly dependent on the performance of the resynchronization technique. Basically, if the resynchronization method can efficiently localize the damaged data area, the error concealment strategy becomes much more tractable. Error concealment is actually a decoder issue if there is no additional information provided by the encoder. There are many approaches to error concealment, which are referred to in Chapter 17.

18.4 MPEG-4 VISUAL BITSTREAM SYNTAX AND SEMANTICS

The common feature of MPEG-4 and MPEG-1/MPEG-2 is the layered structure of the bitstream. MPEG-4 defines a syntactic description language to describe the exact binary syntax of an audio-visual object bitstream, as well as that of the scene description information. This provides a consistent and uniform way to describe the syntax in a very precise form, while at the same time simplifying bitstream compliance testing. The visual syntax hierarchy includes the following layers:

- Video session (VS)
- Video object (VO)
- Video object layer (VOL) or texture object layer (TOL)
- Group of video object plane (GOV)
- Video object plane (VOP)

A typical video syntax hierarchy is shown in Figure 18.13.

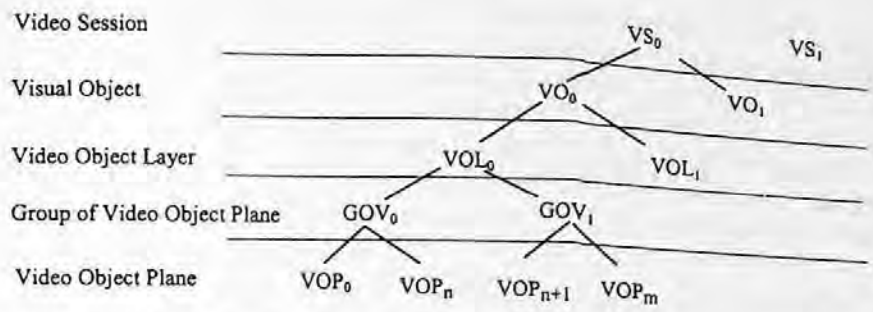


FIGURE 18.13 MPEG-4 video syntax hierarchy.

The video session (VS) is the highest syntactic structure of the coded video bitstream. A VS is a collection of one or more VOs. A VO can consist of one or more layers. Since MPEG-4 is extended from video coding to visual coding, the type of visual objects not only includes video objects, but also still texture objects, mesh objects, and face objects. These layers can be either video or texture. Still texture coding is designed for high-visual-quality applications in transmission and rendering of texture. The still coding algorithm supports a scalable representation of image or synthetic scene data such as luminance, color, and shape. This is very useful for progressive transmission of images or 2-D/3-D synthetic scenes. The images can be gradually built up in the terminal monitor as they are received. The bitstreams for coded mesh objects are nonscalable; they define the structure and motion of a 2-D mesh. The texture of the mesh has to be coded as a separate video object. The bitstreams for face objects are also nonscalable; these bitstreams contain the face animation parameters. VOs are coded with different types of scalability. The base layer can be decoded independently and the enhancement layers can only be decoded with the base layer. In the special case of a single rectangular VO, all of the MPEG-4 layers can be related to MPEG-2 layers. That is, VS is the same as VO since in this case a single VO is a video sequence, VOL or TOL is the same as the sequence scalable extension, GOV is like the GOP, and VOP is a video frame. VO sequence may contain one or more VOs coded concurrently. The VO header information contains the start code followed by profile and level identification and a VO identification to indicate the type of object, which may be a VO, a still texture object, a mesh object, or a face object. The VO may contain one or more VOLs. In the VOL, the VO can be coded with spatial or temporal scalability. Also, the VO may be encoded in several layers from coarse to fine resolution. Depending on the application need, the decoder can choose the number of layers to decode. A VO at a specified time is called a video object plane (VOP). Thus, a VO contains many VOPs. A scene may contain many VOs. Each VO can be encoded to an independent bitstream. A collection of VOPs in a VOL is called a group of VOPs (GOV). This concept corresponds to the group of pictures (GOP) in MPEG-1 and MPEG-2. A VOP is then coded by shape coding and texture coding, which is specified at lower layers of syntax, such as the macroblock and block layer. The VOP or higher-than-VOP layer always commences with a start code and is followed by the data of lower layers, which is similar to the MPEG-1 and MPEG-2 syntax.

18.5 MPEG-4 VIDEO VERIFICATION MODEL

Since all video-coding standards define only the bitstream syntax and decoding process, the use of test models to verify and optimize the algorithms is needed during the development process. For this purpose a common platform with a precise definition of encoding and decoding algorithms has to be provided. The test model (TM) of MPEG-2 took the above-mentioned role. The TM of MPEG-2 was updated continually from version 1.0 to version 5.0, until the MPEG-2 Video IS (International Standard) was completed. MPEG-4 video uses a similar tool during the development process; this tool in MPEG-4 is called the Verification Model (VM). So far, the MPEG-4 video VM has gradually evolved from version 1.0 to version 12.0 and in the process has addressed an increasing number of desired functionalities such as content-based scalability, error resilience, coding efficiency, and so on. The material presented in this section is different from Section 18.3. Section 18.3 presented the technologies adopted or that will be adopted by MPEG-4, while this section provides an example of how to use the standard, for example, how to encode or generate the MPEG-4-compliant bitstream. Of course, the decoder is also included in the VM.

18.5.1 VOP-Based Encoding and Decoding Process

Since the most important feature of MPEG-4 is an object-based coding method, the input video sequence is first decomposed into separate VOs, these VOs are then encoded into separate bitstreams so that the user can access and manipulate (cut, paste, etc.) the video sequence in the bitstream domain. Instances of VOs in a given time are called a video object plane (VOP). The bitstream also contains the composition information to indicate where and when each VOP is to be displayed. At the decoder, the user may be allowed to change the composition of the scene displayed by interactively changing the composition information.

18.5.2 Video Encoder

For object-based coding, the encoder consists mainly of two parts: the shape coding and the texture coding of the input VOP. The texture coding is based on the DCT coding with traditional motion-compensated predictive coding. The VOP is represented by means of a bounding rectangular as described previously. The phase between luminance and chrominance pixels of the bounding rectangular has to be correctly set to the 4:2:0 format as in MPEG-1/2. The block diagram of encoding structure is shown in Figure 18.14.

The core technologies used in VOP coding of MPEG-4 have been described previously. Here we are going to discuss several encoding issues. Although these issues are essential to the performance and application, they are not dependent on the syntax. As a result, they are not included as normative parts of the standard, but are included as informative annexes.

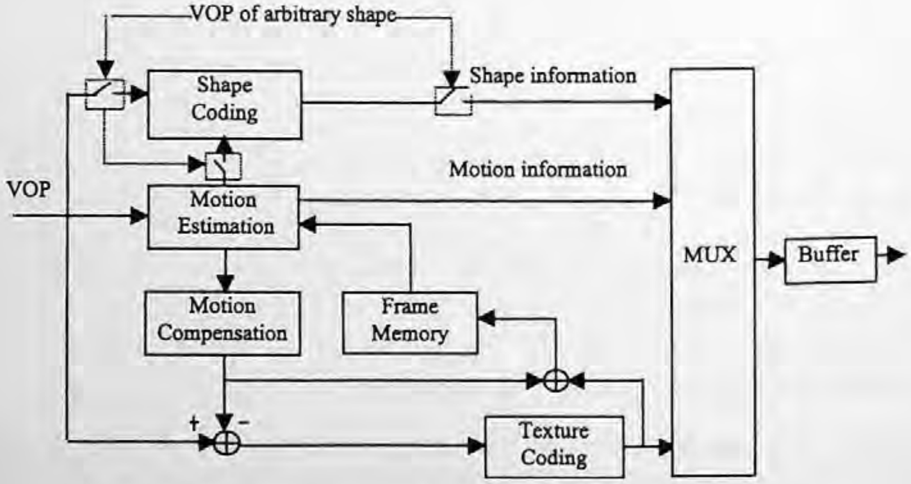


FIGURE 18.14 Block diagram of MPEG-4 video encoder structure.

18.5.2.1 Video Segmentation

Object-based coding is the most important feature of MPEG-4. Therefore, the tool for object boundary detection or segmentation is a key issue in efficiently performing the object-based coding scheme. But the method of decomposing a natural scene to several separate objects is not specified by the standard since it is a preprocessing issue. There are currently two kinds of algorithms for segmentation of video objects. One kind of algorithm is an automatic segmentation algorithm. In the case of real-time applications, the segmentation must be done automatically. Real-time automatic segmentation algorithms are currently not mature. An automatic segmentation algorithm has been proposed in MPEG96/M960 (Colonnese and Russo, 1996). This algorithm separates regions corresponding to moving objects from regions belonging to a static background for each frame of a video sequence. The algorithm is based on a motion analysis for each frame. The motion analysis is performed along several frames to track each pixel of the current frame and to detect whether the pixel belongs to the moving objects.

Another kind of segmentation algorithms is one that is user assisted or "semiautomatic." In non-real-time applications, the semiautomatic segmentation may be used effectively and give better results than the automatic segmentation. In the core experiments of MPEG-4, a semiautomatic segmentation algorithm was proposed in MPEG97/M3147 (Choi et al., 1997). The block diagram of the semiautomatic segmentation is shown in Figure 18.15.

This technique consists of two steps. First, the intraframe segmentation is applied to the first frame, which is considered as a frame that either contains newly appeared objects or a reset frame. Then the interframe segmentation is applied to the consecutive frames. For intraframe, the segmentation is processed by a user manually or semiautomatically. The user uses a graphical user interface ·(GUI) to draw the boundaries of objects of interest. The user can mask the entire objects all the way around objects using a mouse with a predefined thickness of the line (number of pixels). A marked swath is then achieved by the mouse, and this marked area is assumed to contain the object boundaries. A boundary-detection algorithm is applied to the marked area to create the real object boundaries. For interframe segmentation, an object boundary-tracking algorithm is proposed to obtain the object boundaries of the consecutive frames. At first, the boundary of the previous object is extracted and the motion estimation is performed on the object boundary. The object boundary

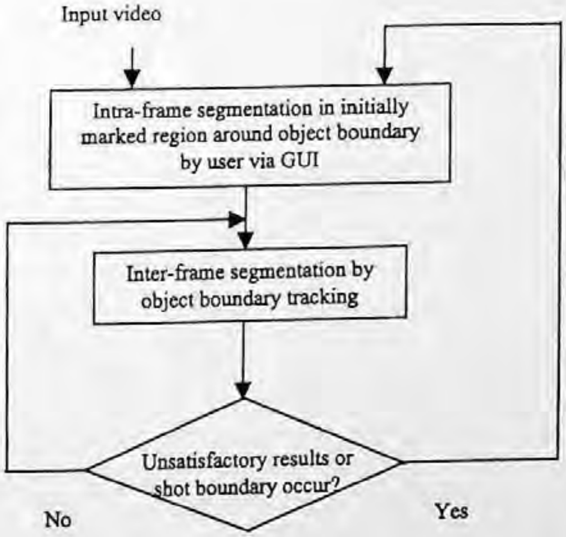

Input video
Intra-frame segmentation in initially marked region around object boundary by user via GUI
Inter-frame segmentation by object boundary tracking
Unsatisfactory results or shot boundary occur?
No
Yes

FIGURE 18.15 Block diagram of a user-assisted VO segmentation method.

of the current frame is initially obtained by motion compensation and then refined by using temporal information and spatial information all the way around the object boundary. Finally, the refined object boundary can be obtained. As mentioned previously, the segmentation technique is an important tool for object-based processing in MPEG-4, but it is not defined by the standard. The method described here is just an example provided by the core experiments of MPEG-4. There are many other algorithms under investigation, such as the circular Viterbi algorithm described by Lin et al. (1998).

18.5.2.2 Intra/Intermode Decision

For inter-VOP coding, a macroblock can be coded in one of four modes. These four modes include direct coding mode, forward coding, backward coding, and bidirectional coding. In the encoder we have to decide which mode is the best. The mode decision is an important part of encoding optimization. An example of the selection of an optimized mode decision has been given in Chapter 17 for an MPEG-2 encoder. The same technique can be extended to an MPEG-4 encoder. The basic idea of mode decision is to choose the coding mode that results in the best operation point on the rate–distortion curve. For obtaining the best operation point on the rate–distortion curve, the encoder has to compare all possible coding modes and choose the best one. This is a very complicated procedure. In the MPEG-2 case, we used a quadratic model to unify the measures of bits used to code prediction residues and the motion vectors. A simplified mode but near-optimized mode decision method has resulted. Here, the VM.12 proposes the following steps to make coding mode decisions. First, the motion-compensated prediction error is calculated by each of the four modes. Next, the SAD of each of the motion-compensated prediction macroblocks is calculated and compared with the variance of the macroblock to be coded. Then, a mode of generating the smallest SAD (for direct mode, a bias is applied) is selected. For the interlaced video, more coding modes are involved. This method of mode decision is simple, but it is not optimal since the cost for coding motion vectors is not considered. Consequently, the mode may not lie on the best operation point on the distortion curve. But again, this is an encoding issue; the encoding designers have the freedom to use their own algorithm. The VM just provides an example of an encoder that can generate the compliant bitstream.

18.5.2.3 Off-Line Sprite Generation

The sprite is a useful tool in MPEG-4 for coding a certain kind of video sequences at very low bit rates. The method of generating a sprite for a video sequence is an encoder issue. The VM gives an example of off-line sprite generation. For a natural VO, a sprite is referred to as a representative view collected from a video sequence. Before decoding, the sprite is transmitted to the decoder. Then the motion compensation can be performed by using the sprite from which the video can be reconstructed. The effectiveness of video reconstruction depends on whether the motion of the object can be effectively represented by a global motion model such as translation, zooming, affine, and perspective. The key technology of the sprite generation is the motion estimation to find perspective motion parameters. This can be implemented by many algorithms described in this book such as the three-step matching technique. The block diagram of sprite generation using the perspective motion estimation is shown as in Figure 18.16.

The sprite is generated from the input video sequence by the following steps. First, the first frame is used as the initial value of sprite. From the second frame, the motion estimation is applied to find the perspective motion parameters between two frames. The current frame is wrapped toward the initial sprite using the perspective motion vectors to get wrapped image. Then the wrapped image is blended with initial sprite to obtain a updated sprite. This procedure is continued to the entire video sequence. The final sprite is then generated.

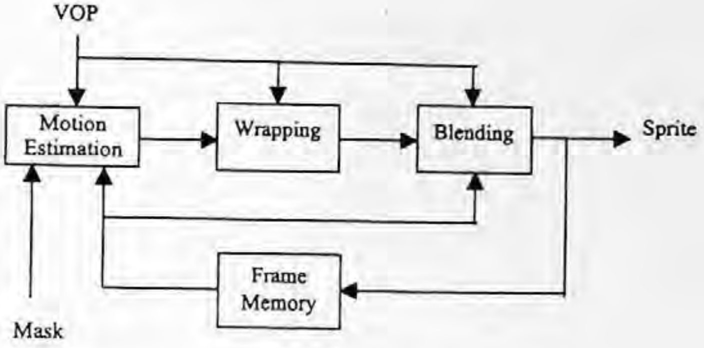


FIGURE 18.16 Block diagram of sprite generation.

18.5.2.4 Multiple VO Rate Control

As we know, the purpose of rate control is to obtain the best coding performance for a given bit rate in constant-bit-rate video coding. In MPEG-4 video coding, there is an additional objective for rate control: how to assign the bits among multiple VOs. In the multiple VO video coding rate control algorithm, the total target is first adjusted based on the buffer fullness, and then distributed proportional to the size of the object, the motion which the object is experiencing, and its maximum absolute differences. Based on the new individual targets and second-order model parameters (Lee et al., 1997), appropriate quantization parameters can be calculated for each VO. To compromise the trade-offs in spatial and temporal coding, two modes of operation have been introduced. With these modes, suitable decisions can be made to differentiate between low- and high-bit-rate coding. In addition, a shape rate control algorithm has been included. The algorithm for performing the joint rate control can be decomposed into a preencoding stage and a postencoding stage. The preencoding stage consists of (1) the target bit estimation, (2) joint buffer control, (3) pre-frame-skip control, and (4) the quantization level and alpha threshold calculation. The postencoding stage consists of (1) updating the rate–distortion model, (2) post-frame-skip control, and (3) determining the mode of operation. The initialization process is very similar to the single VOP initialization process. Since a single buffer is used, the buffer drain rate and initializations remain the same, but many of the parameters are extended to vector quantities. As a means of regulating the trade-offs between spatial and temporal coding, two modes of operation are introduced: low mode and high mode. When encoding at high bit rates, the availability of bits allows the algorithm to be flexible in its target assignment to each VO. Under these circumstances, it is reasonable to impose homogeneous quality among each VO. Therefore, the inclusion of $MAD^2[i]$ is essential to the target distribution and should carry the highest weighting. On the other hand, when the availability of bits is limited, it is very difficult (if not impossible) to achieve homogeneous quality among the VOs. Under these conditions, it is desirable to spend fewer bits on the background and more bits on the foreground. Consequently, the significance of the variance has decreased and the significance of the motion has increased. Besides regulating the quality within each frame, it is also important to regulate the temporal quality as well, i.e., to keep the frame skipping to a minimum. In high mode, this is very easy to do since the availability of bits is plentiful. However, in low mode, frame-skipping occurs much more often. In fact, the number of frames being skipped is a good indication in which mode the algorithm should be operating. Overall, this particular algorithm is able to achieve the target bit rate successfully, effectively code arbitrarily shaped objects, and maintain a stable buffer (Vetro et al., 1999).

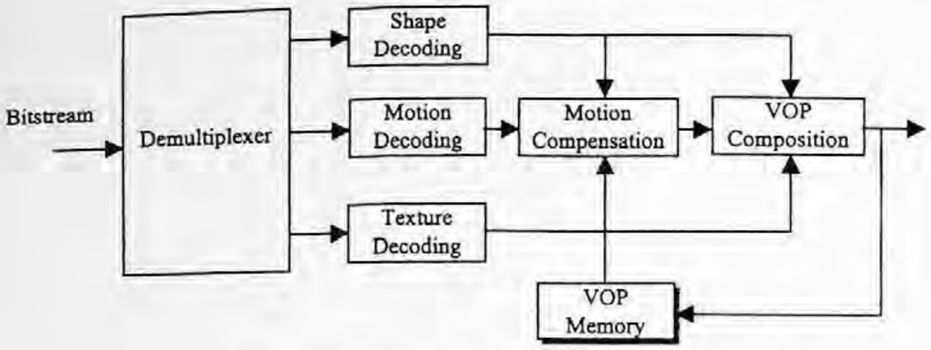


FIGURE 18.17 VOP decoder structure.

18.5.3 Video Decoder

The decoder mainly consists of three parts: shape, motion, and texture decoding. The decoder block diagram is shown in Figure 18.17. At the decoder the bitstream is first demultiplexed into shape information and motion information as well as texture information. The reconstructed VOP is obtained by the right combination of the shape, texture, and motion information. The shape decoding is a unique feature of the MPEG-4 decoder. The basic technology of shape decoding is context-based arithmetic decoding and block-based motion compensation.

The primary data structure is denoted is the binary alpha block (BAB). The BAB is a square block of binary pixels representing the opacity or transparency for the pixels in a specified block-shaped spatial region of size 16 × 16 pixels which is colocated with each texture macroblock. The block diagram of a texture decoder is shown in Figure 18.18.

Texture decoding is similar to the video decoder in MPEG-1/2 except for inverse DC/AC prediction and more quantization methods. The DC prediction is different from the one used in MPEG-1/2. In MPEG-4 the DC coefficient is adaptively predicted from the above block or left block. The AC prediction is similar to the one used in H.263 but is not used in the MPEG-1/2. For motion compensation, the motion vectors must be decoded. The horizontal and vertical motion vector components are decoded differentially by using a prediction from the spatial neighborhood consisting of three motion vectors already decoded. The final motion vector is obtained by adding the prediction motion vector values to the decoded differential motion values. Also, in MPEG-4 video coding the several advanced motion compensation modes, such as four 8 × 8 motion vector compensation and overlapped motion compensation, have to be handled. Another issue of motion compensation in MPEG-4 is raised by VOP-based coding. To perform motion-compensated prediction on a VOP basis, a special padding technique is used for each of macroblock that lies on the shape boundary of the VOP. The padding process defines the values of pixels, which are located outside the VOP for prediction of arbitrarily shaped objects. Padding for luminance pixels and chrominance pixels is defined in the standard (ISO/IEC, 1998a). The additional decoding issues

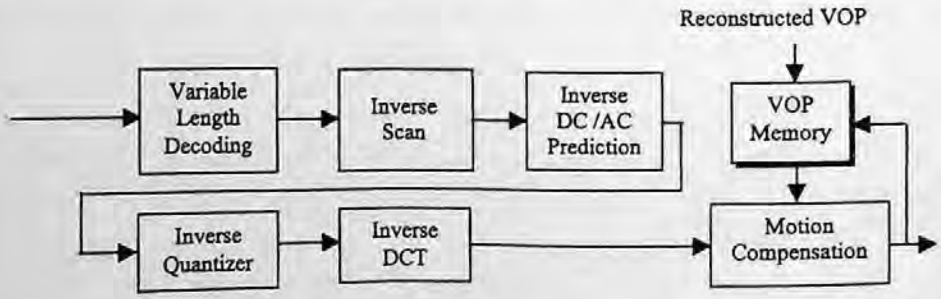


FIGURE 18.18 Block diagram of texture decoding.

which are special for MPEG-4 include sprite decoding, generalized scalable decoding, and still texture decoding. We do not go into further detail for these topics. Interested readers can get detail from the standard documents. The outputs of decoded results are the reconstructed VOPs that are finally sent to the compositor. In the compositor, the VOPs are recursively blended in the order specified by the VOP composition order. It should be noted that the decoders could take advantage of object-based decoding. They are able to be flexible in the composition of the reconstructed VOPs such as reallocating, rotation, or other editing actions.

18.6 SUMMARY

In this chapter, the new video-coding standard, MPEG-4 is introduced. The unique feature of MPEG-4 video is content-based coding. This feature allows the MPEG-4 to provide much functionality, which other video-coding standards do not have. The key technologies used in MPEG-4 video are described. These technologies provide basic tools for MPEG-4 video to provide object-based coding functionality. Finally, the video verification model, a platform of MPEG-4 development and an encoding and decoding example, is described.

18.7 EXERCISES

18-1. Why is object- or content-based coding the most important feature of MPEG-4 visual coding standard? Describe several applications for this feature.

18-2. What are the new coding tools in MPEG-4 visual coding that are different from MPEG-2 video coding? Is MPEG-4 backward compatible with MPEG-2?

18-3. MPEG-4 video coding has the feature of using either a 16 × 16 block motion vector or an 8 × 8 block motion vector. For what kind of video sequences will the 8 × 8 block motion increase coding efficiency? For what kind of video sequences will the 8 × 8 block motion compensation decrease the coding efficiency?

18-4. What approaches for error resilience are supported by the MPEG-4 syntax? Make a comparison with the error resilience method adopted in MPEG-2 (supported by MPEG-2 syntax), and indicate their relative advantages and disadvantages.

18-5. Design an arithmetic coder for zerotree coding and write a program to test it with several images.

18-6. The Sprite is a new feature of MPEG-4 video coding. MPEG-4 specifies the syntax for sprite coding, but does not give any detail about how to generate a sprite. Conduct a project to generate an off-line sprite for a video sequence and use it for coding the video sequence. Do you observe any increased coding efficiency? When do you expect to see such an increase?

18-7. Shape coding (binary-shape coding) is an important part of MPEG-4 due to object-based coding. Besides the shape coding method used in MPEG-4, name another shape coding method. Conduct a project to compare the method you know with the method proposed in MPEG-4. (Do not expect to get better performance, but expect to reduce the complexity.)

REFERENCES

Choi, J. G., M. Kim, H. Lee, and C. Ahn, Partial experiments on a user-assisted segmentation technique for video object plane generation, MPEG97/M3147, San Jose Meeting of ISO/IEC JTC1/SC29/WG11, Feb. 1998.

Colonnese, S. and G. Russo, FUB results on core experiment N2: comparison of automatic segmentation techniques, MPEG96/M960, Tempere, July 1996.

ISO/IEC 14496-2, Coding of audio-visual objects, part 2, Dec. 18, 1998a.

ISO/IEC 14496-2 Video Verification Model V.12, N2552, December, 1998b.

Lee, H. J., T. Chiang, and Y. Q. Zhang, Scalable rate control for very low bit-rate coding, *Proceedings Internatinoal Conference on Image Processing (ICIP'97)*, II, 768-771, Santa Barbara, CA, Oct. 1997.

Lin, I. J., S. Y. Kung, A. Vetro, and H. Sun, *Circular Viterbi: Boundary Detection with Dynamic Programming*, MPEG, 1998.

Shapiro, J. Embedded image coding using zerotrees of wavelet coefficients, *IEEE Trans. Signal Proc.*, 3445-3462, 1993.

Vetro, A., H. Sun, and Y. Wang, MPEG-4 rate control for multiple video objects, *IEEE Trans. Circ. Syst. Video Technol.*, 9(1), 186-199, 1999.