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

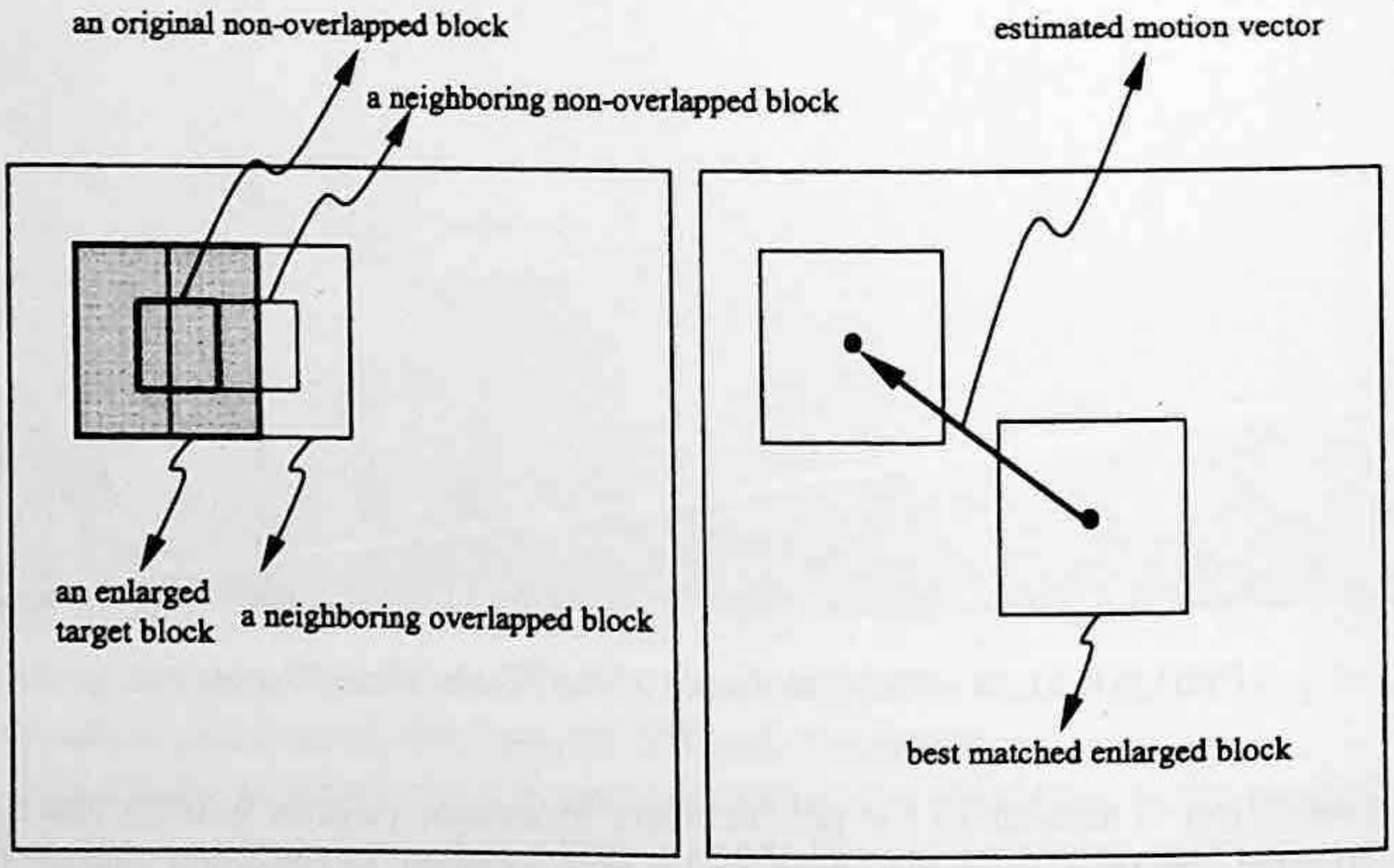


FIGURE 11.21 Overlapped block matching.

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

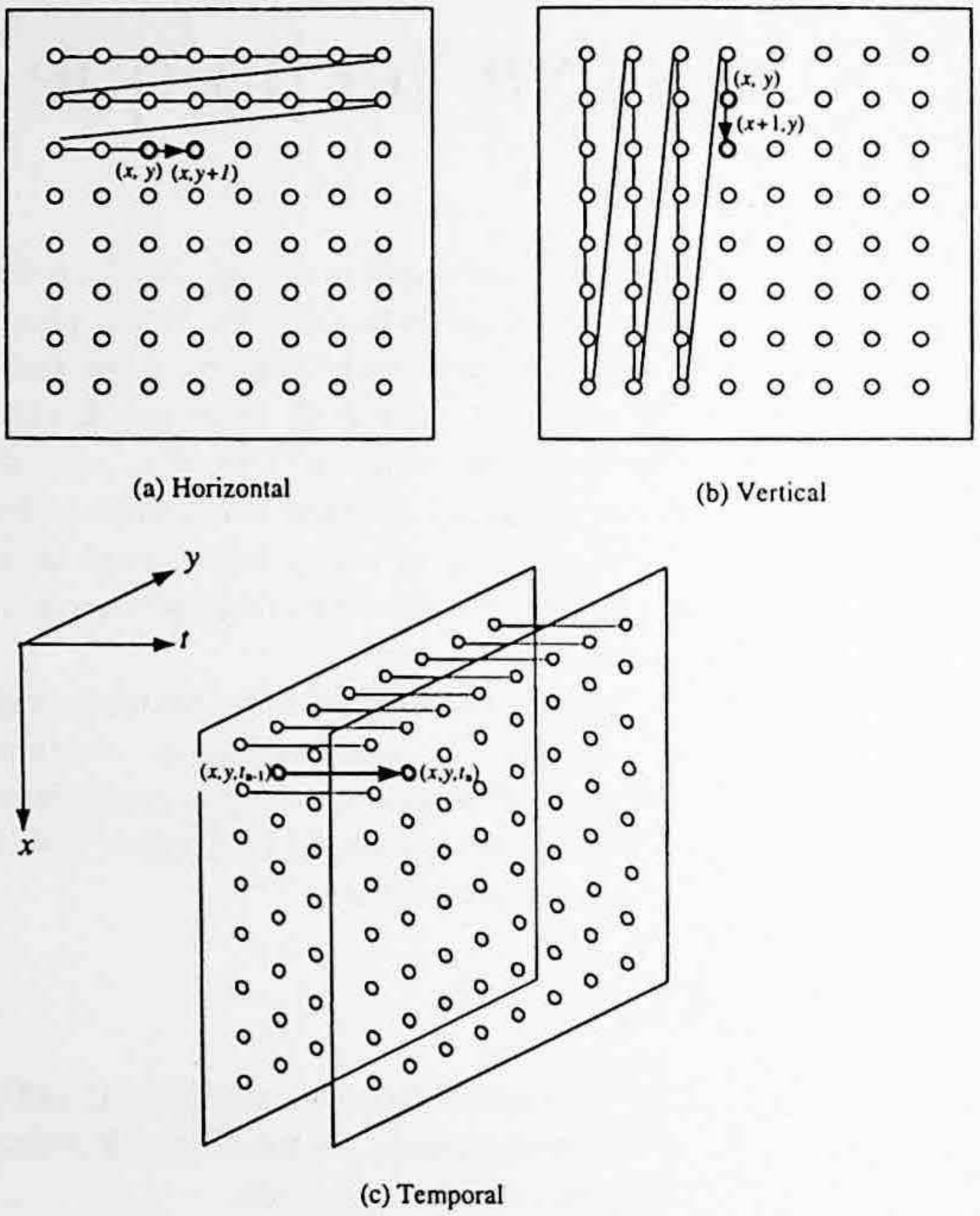
In 1979 Netravali and Robbins published the first pel recursive algorithm to estimate displacement vectors for motion-compensated interframe image coding. Netravali and Robbins (1979) defined a quantity, called the displaced frame difference (DFD), as follows.

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

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

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

and

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

where $\mathbf{H}$ 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 $\mathbf{H}$ 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

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

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

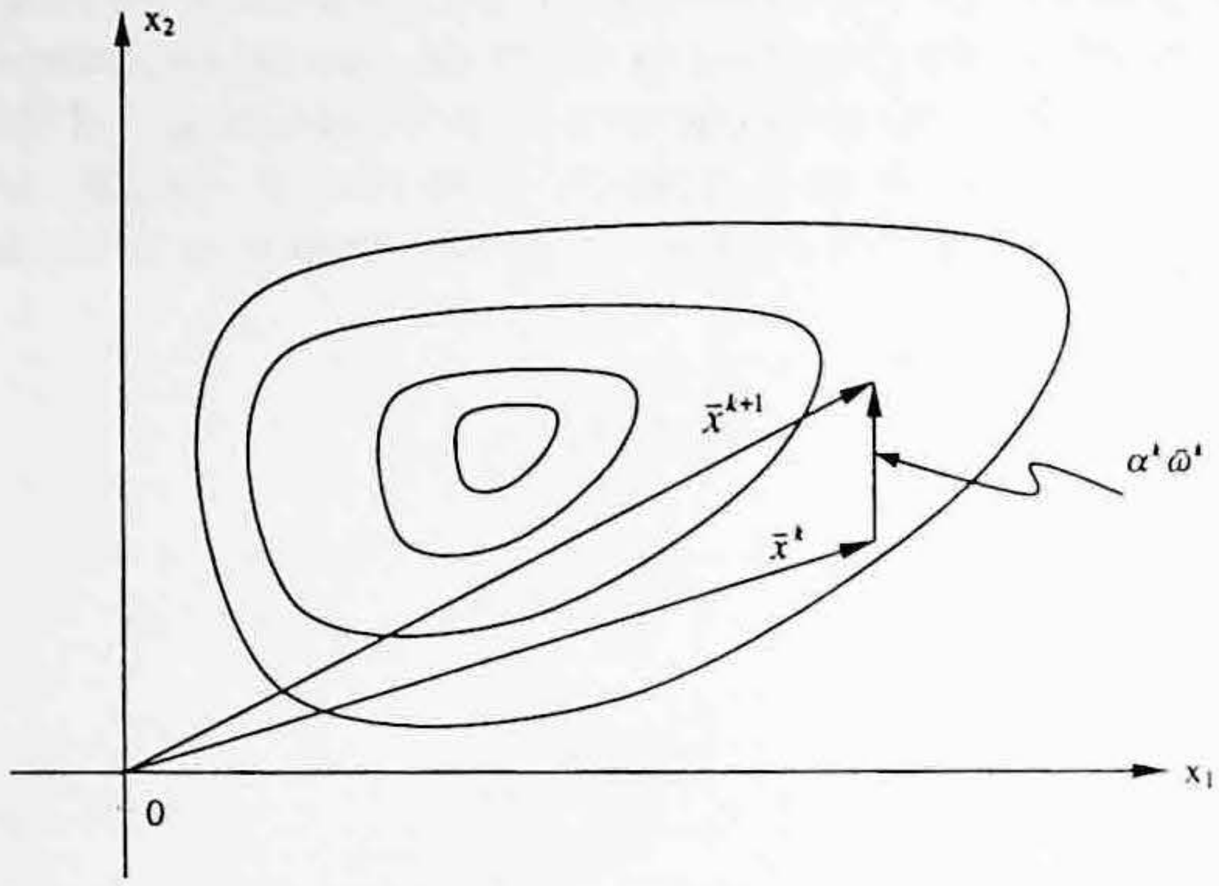


FIGURE 12.2 Descent method.

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

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

where $\langle s, t \rangle$ denotes the inner product between vectors $\bar{s}$ and $\vec{t}$; 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,

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

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

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

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

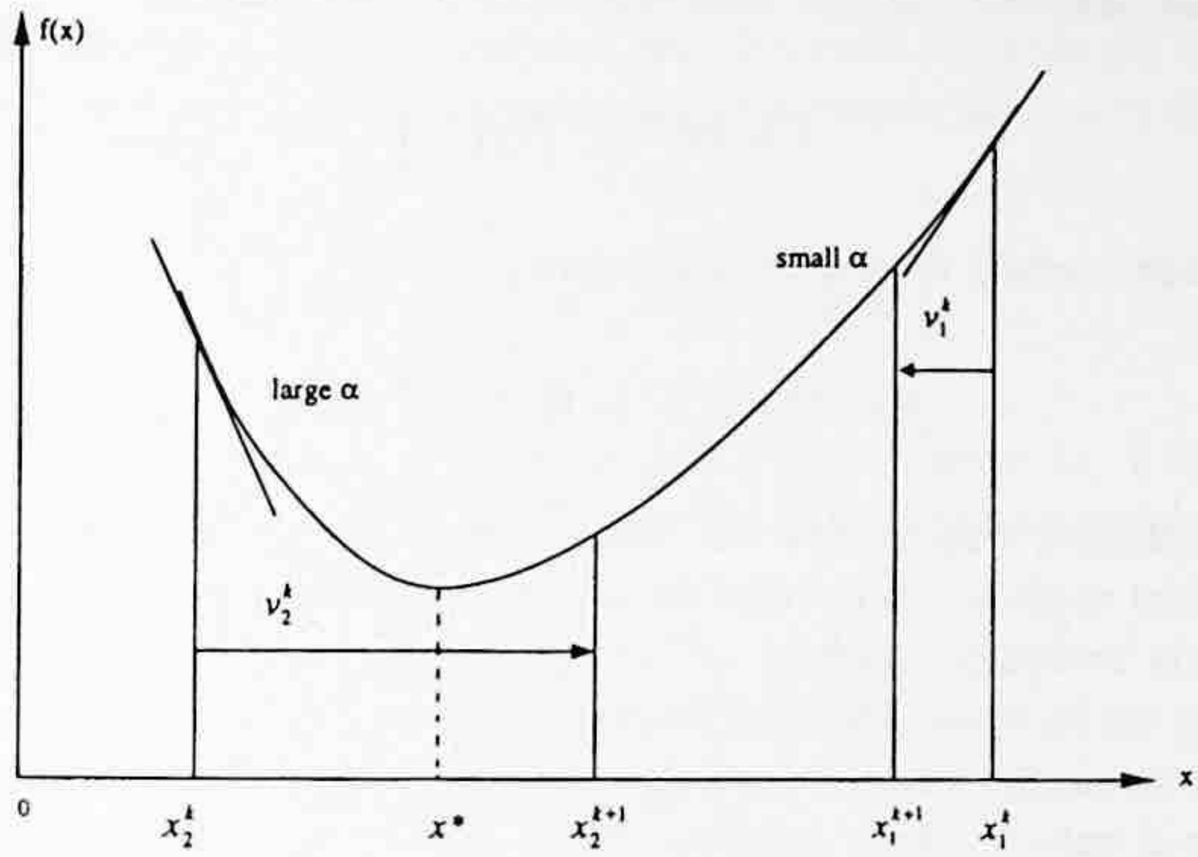


FIGURE 12.3 An illustration of effect of selection of step size on minimization performance. Too small α requires more steps to reach x^*. Too large α may cause overshooting.

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 $k+1$th 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.

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

where φ denotes the higher-order terms, ∇ the gradient, and $\mathbf{H}$ 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

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

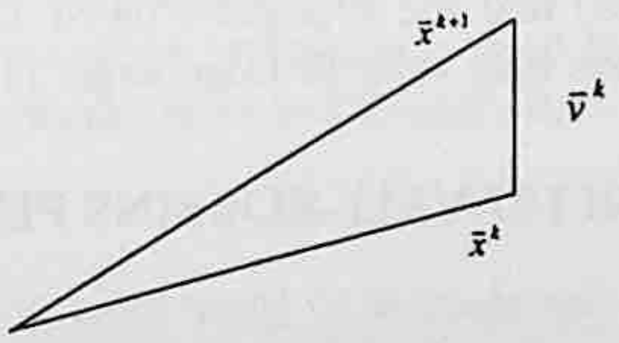


FIGURE 12.4 Derivation of the Newton–Raphson method.

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 $G = [\zeta^k \boldsymbol{I} + \boldsymbol{H}(\bar{x}^k)]^{-1}$ 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,

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

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

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

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

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

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,

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

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:

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

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 $f(\bar{x})$ discussed in Section 12.2 now becomes $DFD^2(x, y, \bar{d})$. The Hessian matrix **H**, consisting of the second-order partial derivatives of the $f(\bar{x})$ with respect to the components of $\bar{x}$ now become the second-order derivatives of DFD^2 with respect to d_x and d_y. Since the vector $\bar{d}$ is a 2-D column vector now, the **H** matrix is hence a 2×2 matrix. That is,

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

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

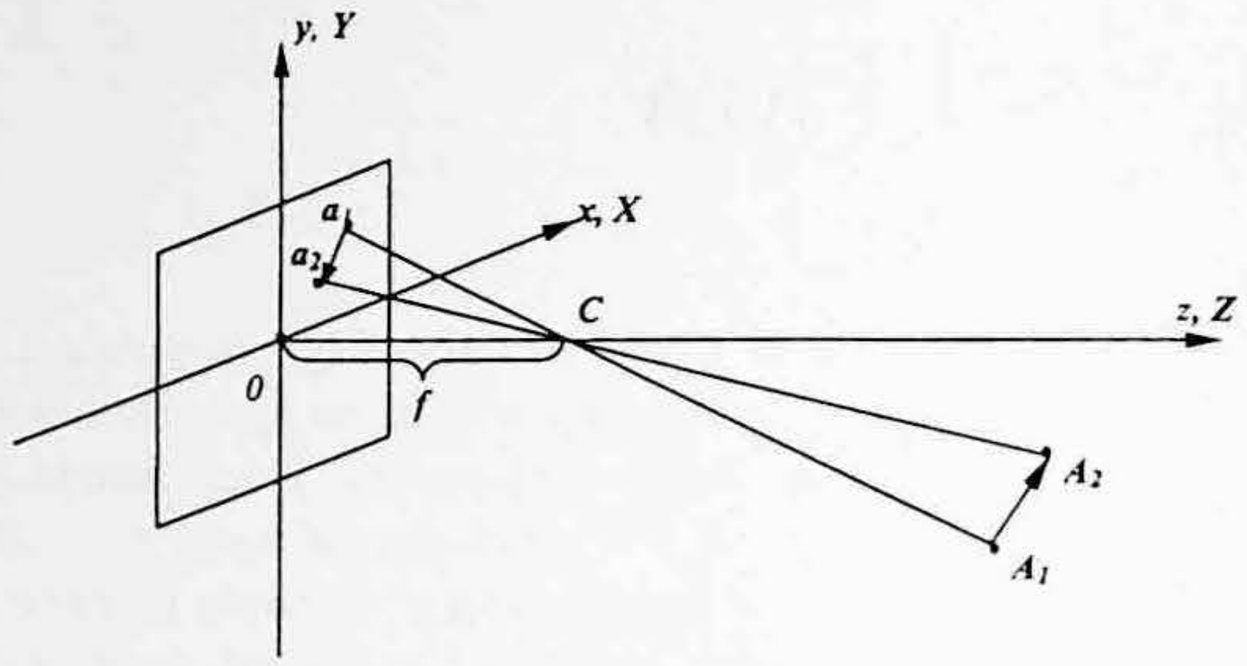


FIGURE 13.1 2-D motion vs. 3-D motion.

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

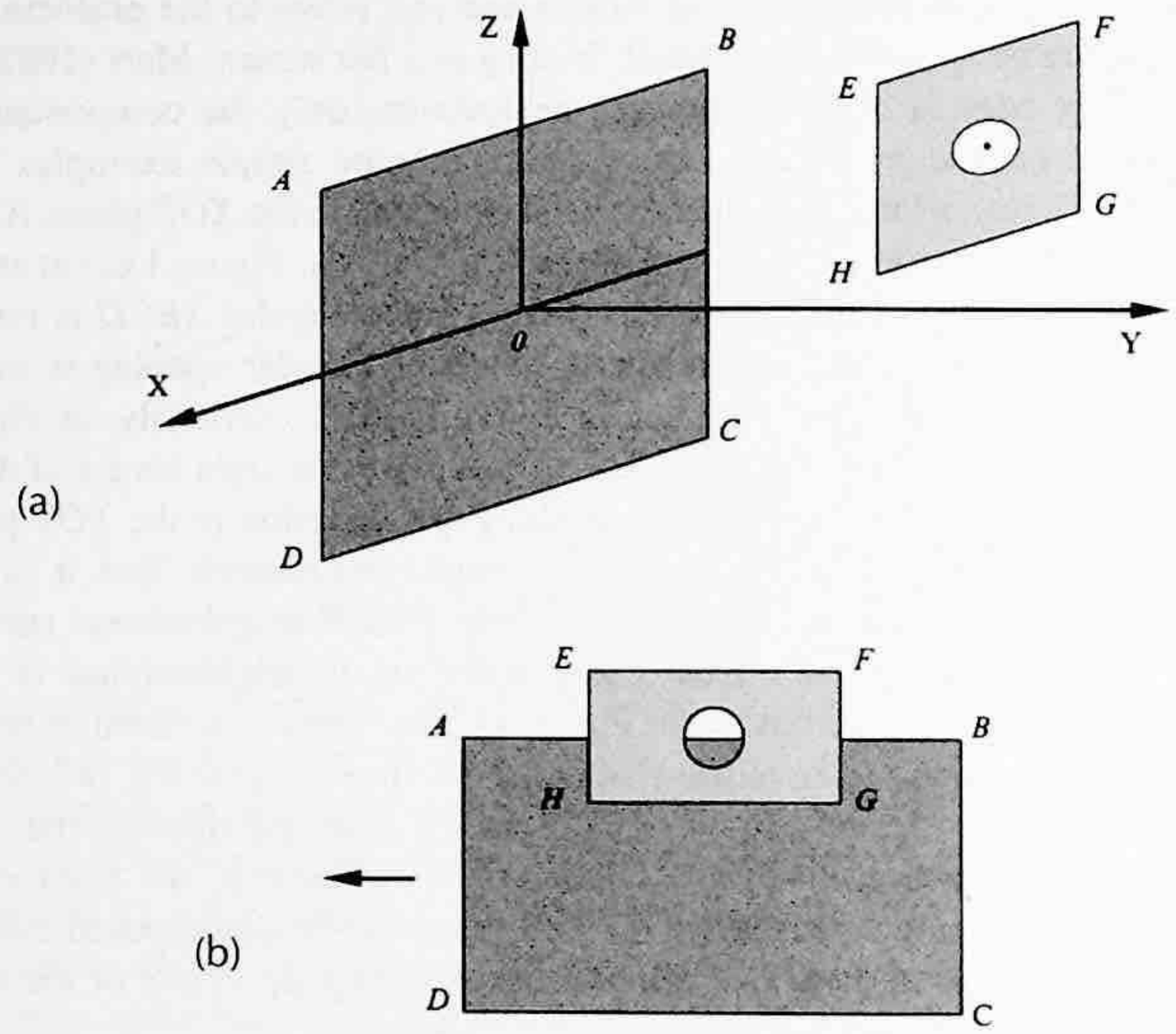


FIGURE 13.2 (a) Aperture problem: A large rectangle ABCD is located in the XOZ plane. A rectangular screen $EFGH$ with a circular aperture is perpendicular to the OY axis. (b) Aperture problem: No motion can be observed through the circular aperture when the rectangular $ABCD$ is moving along the positive 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

$$u = \frac{\partial x}{\partial t}, \quad v \triangleq \frac{\partial y}{\partial t}, \quad u^{\bar{s}} \triangleq \frac{\partial x}{\partial \bar{s}}, \quad u^{\bar{s}} \triangleq \frac{\partial y}{\partial \bar{s}}.$$

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

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

∇^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:

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

The estimation of the partial derivatives of intensity function and the Laplacian of flow vectors need to be addressed. Horn and Schunck considered a $2\times2\times2$ 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

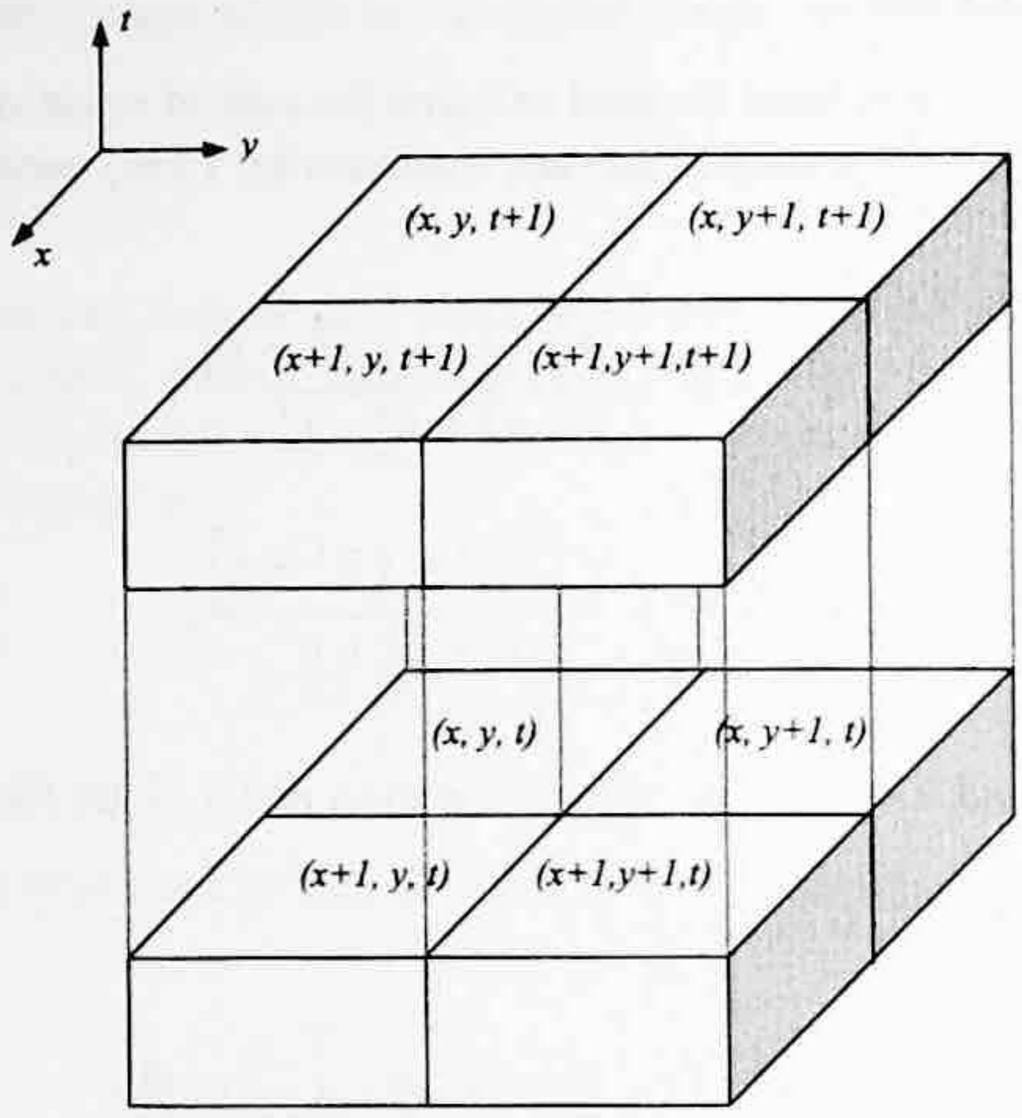
$$\begin{aligned}\nabla^2 u &= \bar{u}(x,y) - u(x,y) \\ \nabla^2 v &= \bar{v}(x,y) - v(x,y).\end{aligned} \tag{13.18}$$

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



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

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

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

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

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

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

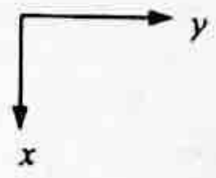
FIGURE 13.3 Estimation of f_x, f_y, and f_t.

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

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

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

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.

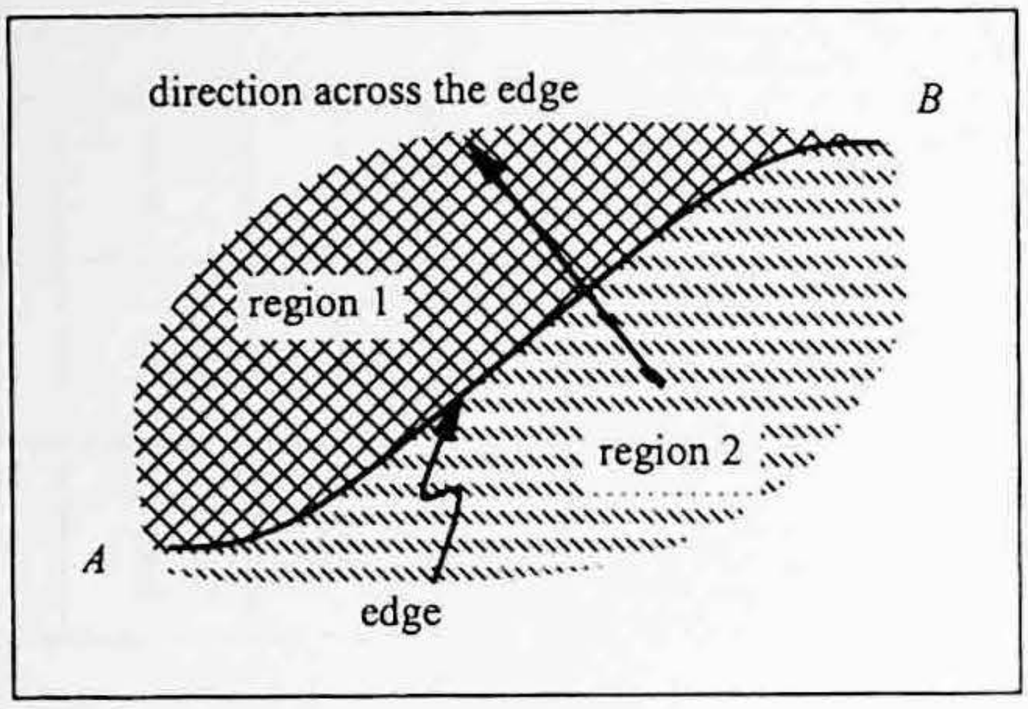


FIGURE 13.6 Oriented-smoothness constraint.

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

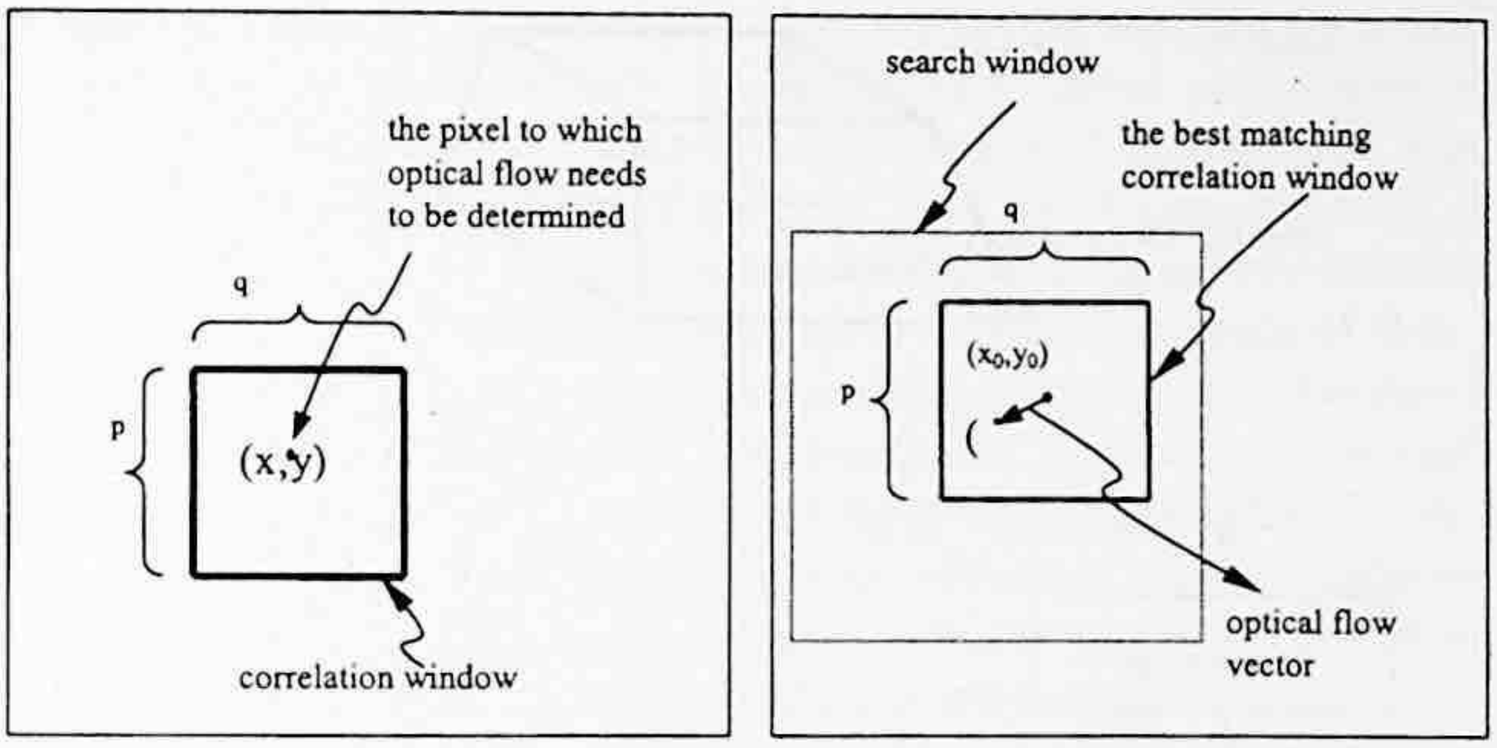


FIGURE 13.7 Correlation-based approach to optical flow determination.

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 $f^k(x, y)$.

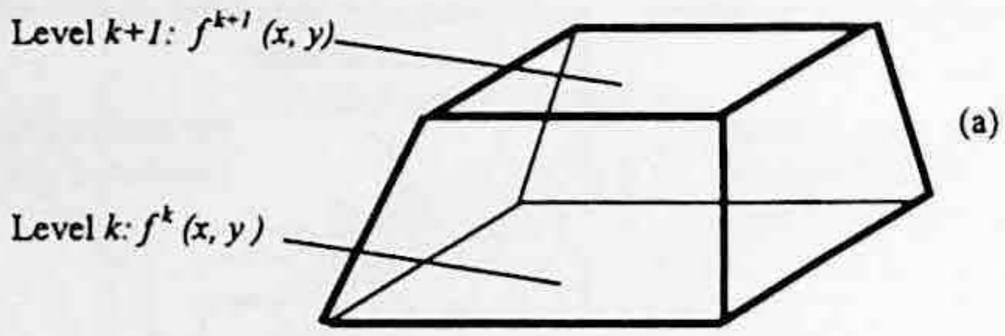


FIGURE 13.8 Laplacian pyramid (level k in a Gaussian pyramid). (a) Two consecutive levels in a pyramid structure. (b) Derivation of level $k + 1$ from level K. (c) Derivation of error at level k in a Laplacian pyramid. (d) Structure of Laplacian pyramid.

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:

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

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

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

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

$$\bar{u} = \frac{\sum_u \sum_v R_n(u,v)u}{\sum_u \sum_v R_n(u,v)}$$

$$\bar{v} = \frac{\sum_u \sum_v R_n(u,v)v}{\sum_u \sum_v R_n(u,v)}, \quad (13.25)$$

and the associated covariance matrix is

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

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

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) 1/16	(0.5×0.25) 1/8	(0.25×0.25) 1/16
(0.5×0.25) 1/8	(0.5×0.5) 1/4	(0.5×0.25) 1/8
(0.25×0.25) 1/16	(0.5×0.25) 1/8	(0.25×0.25) 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 $U=(\bar{u},\bar{v})$, $U_c=(u_c,v_c)$, $U=(u,v)$.

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

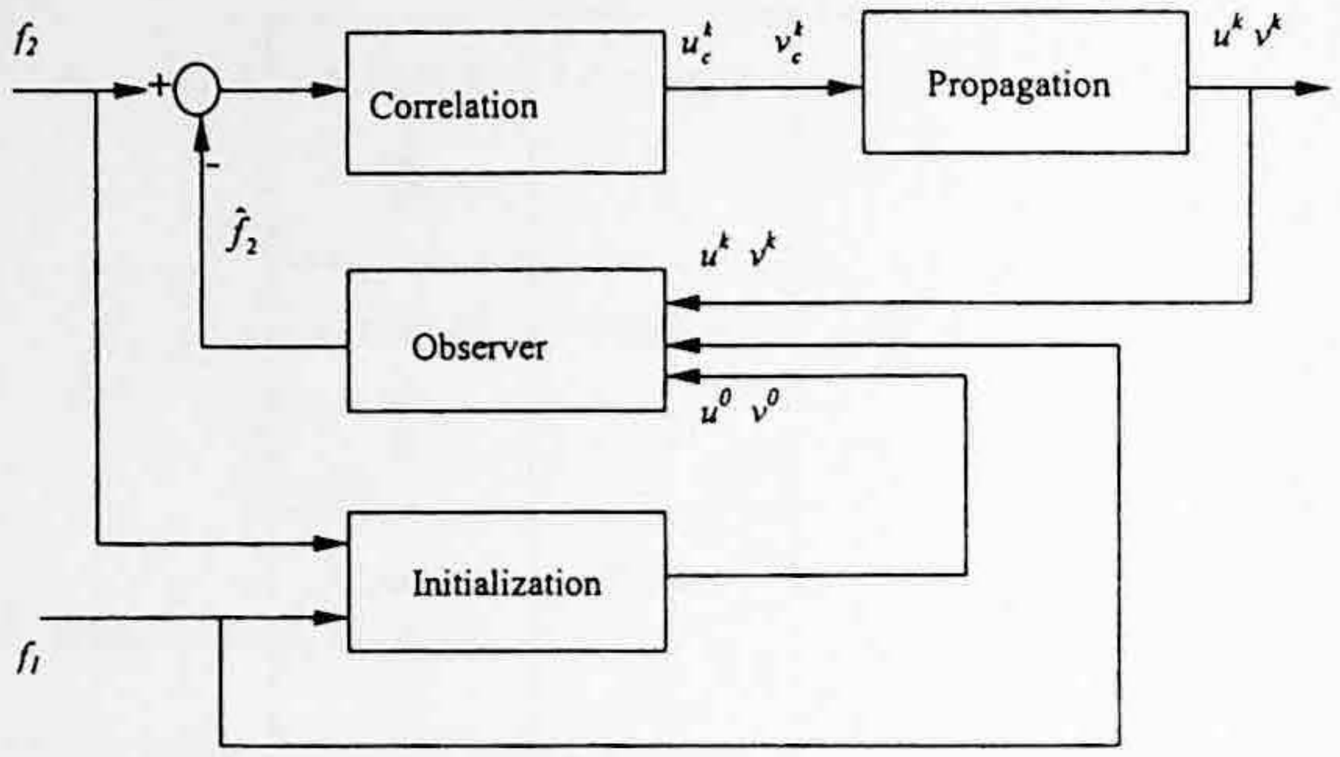


FIGURE 13.10 Block diagram of correlation feedback technique.

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^a, v^a) 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

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

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

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

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.

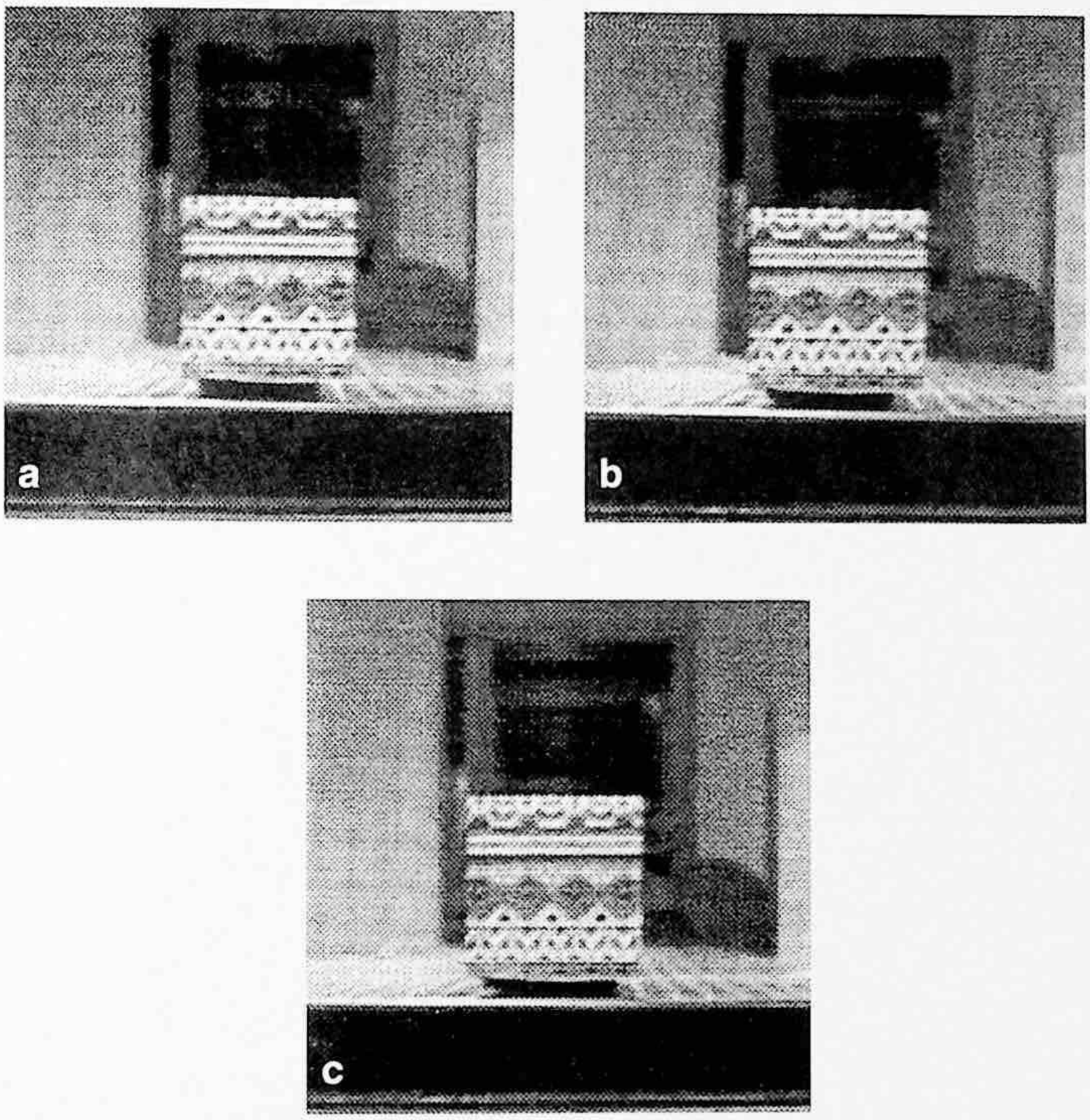
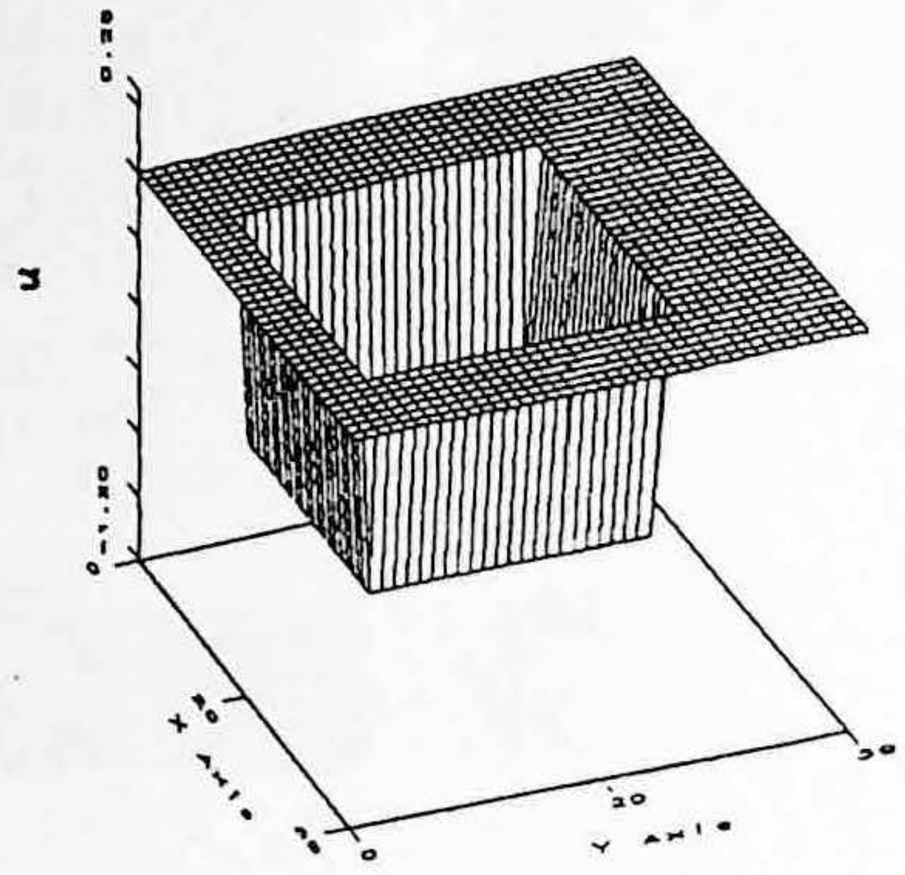


FIGURE 13.11 Texture square (a). Texture square (b). Texture square (c).

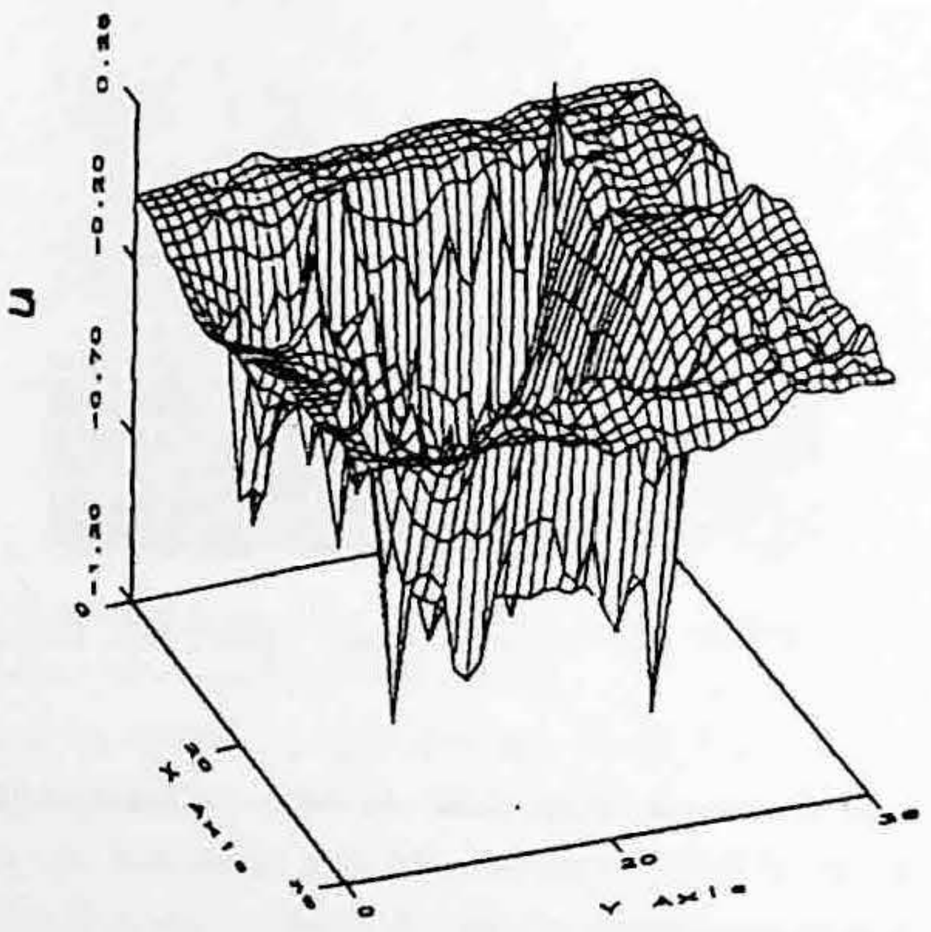
TABLE 13.1
Comparison in Experiment I

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)

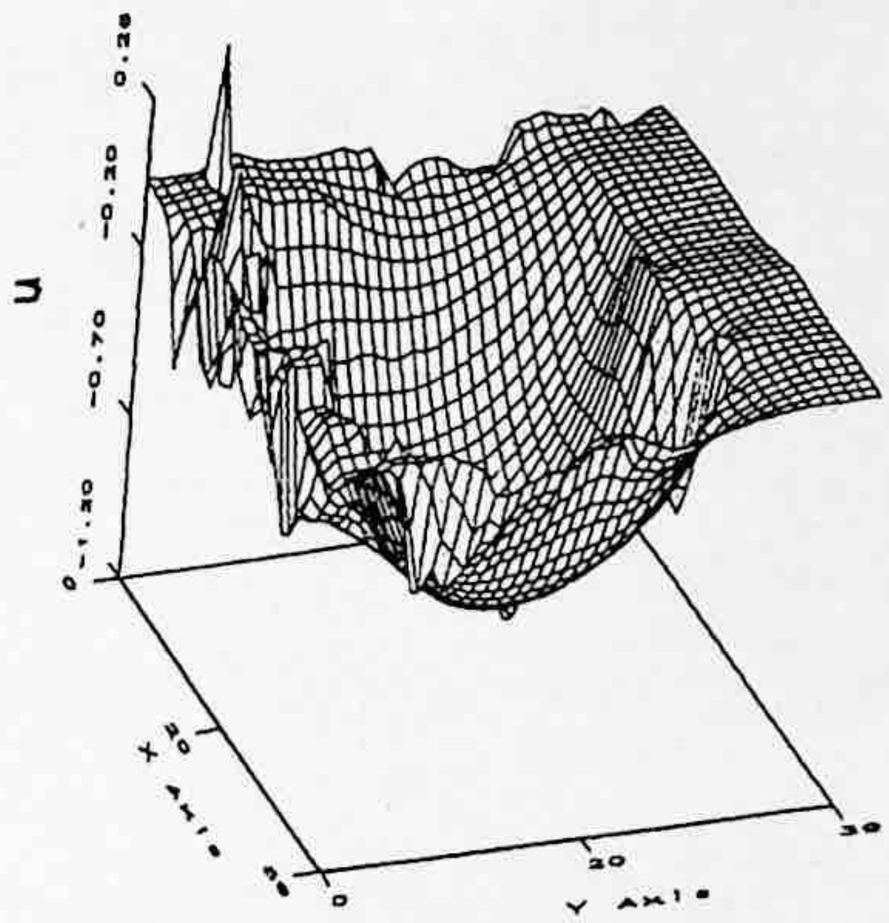


(b)

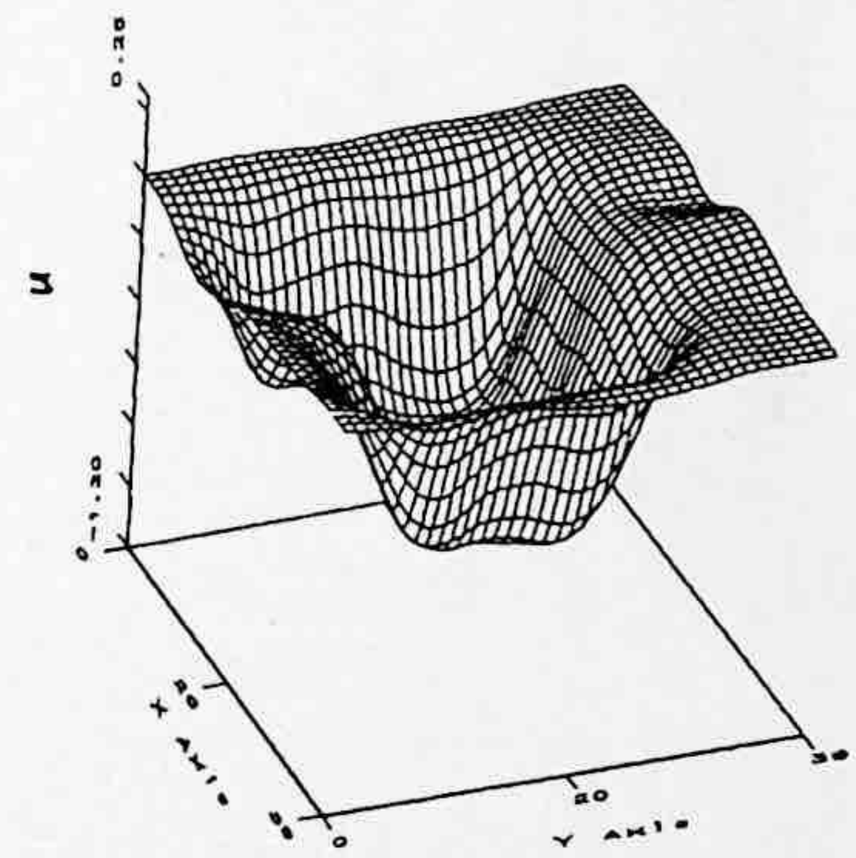
FIGURE 13.12 (a) Correct optical flow field. (b) Optical flow field calculated by the gradient-based approach. (c) Optical flow field calculated by the correlation-based approach. (d) Optical flow field calculated by the correlation-feedback approach.

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,

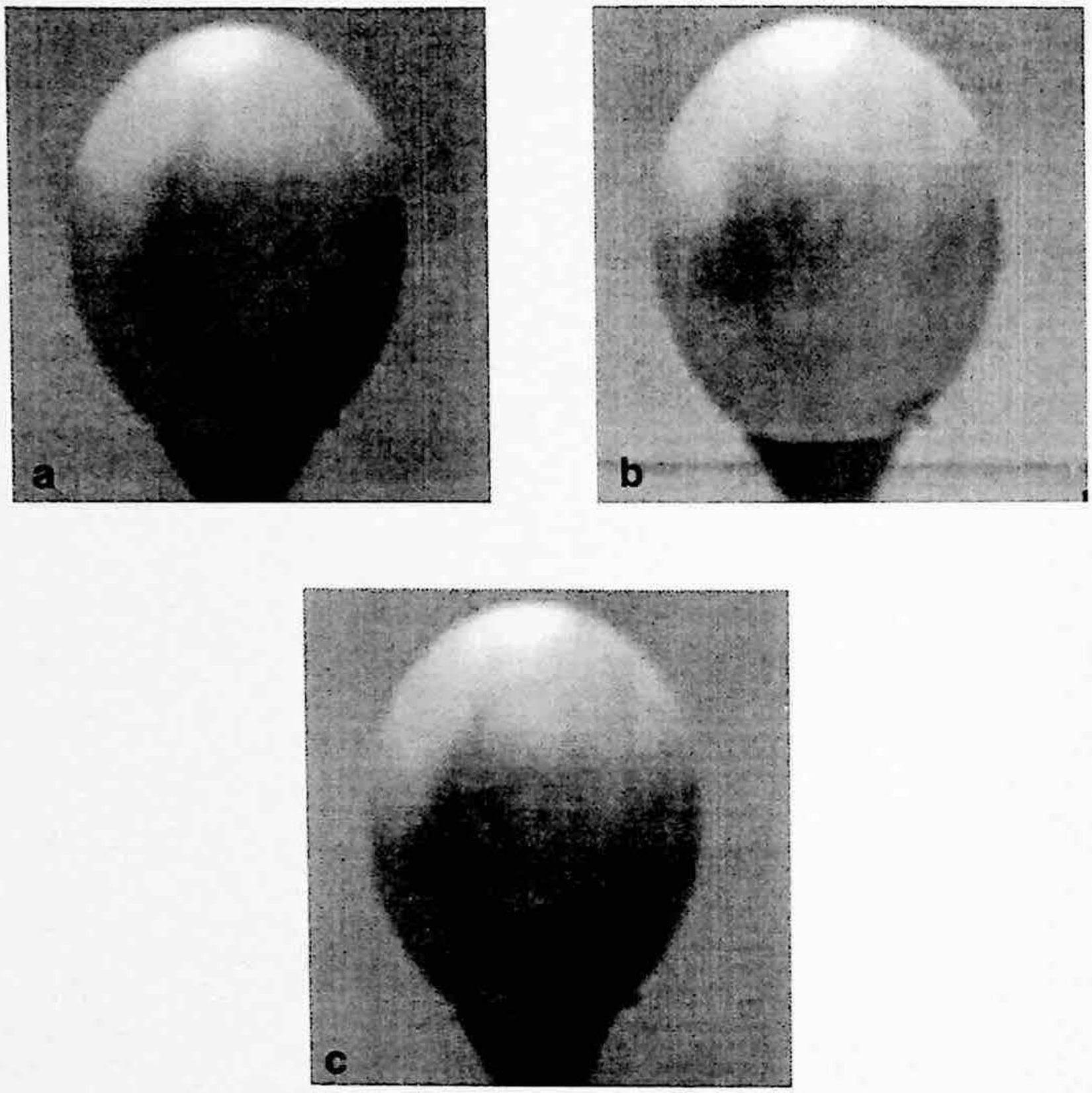


FIGURE 13.13 A rotating ball in three different frames — a, b, c. The rotating velocity is 2.5° per frame.

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 $\psi_E = \text{across}\ (\bar{V}_c \cdot \bar{V}_e)$. 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.

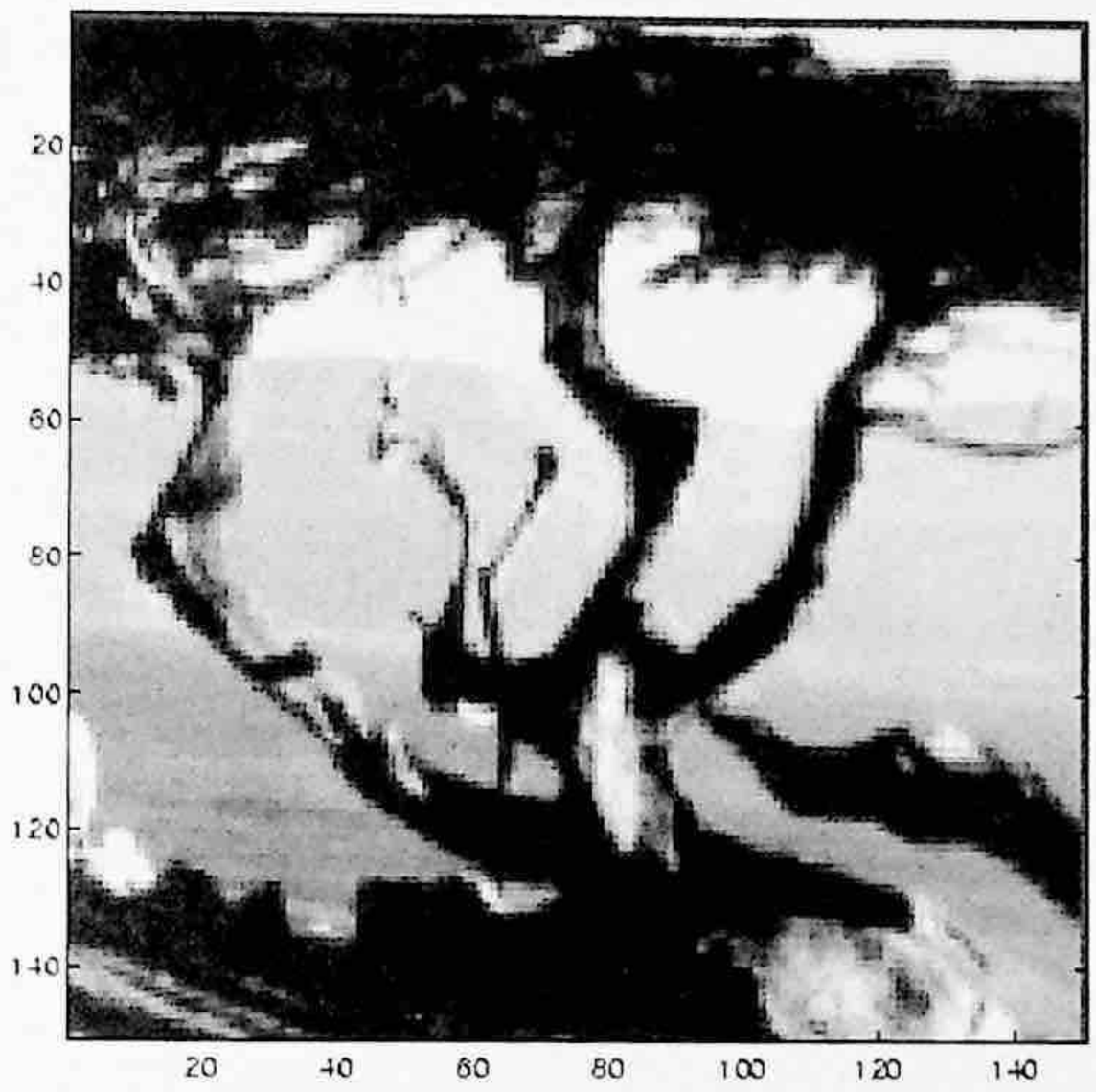


FIGURE 13.14 A frame of the "Tree 2-D" sequence.

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

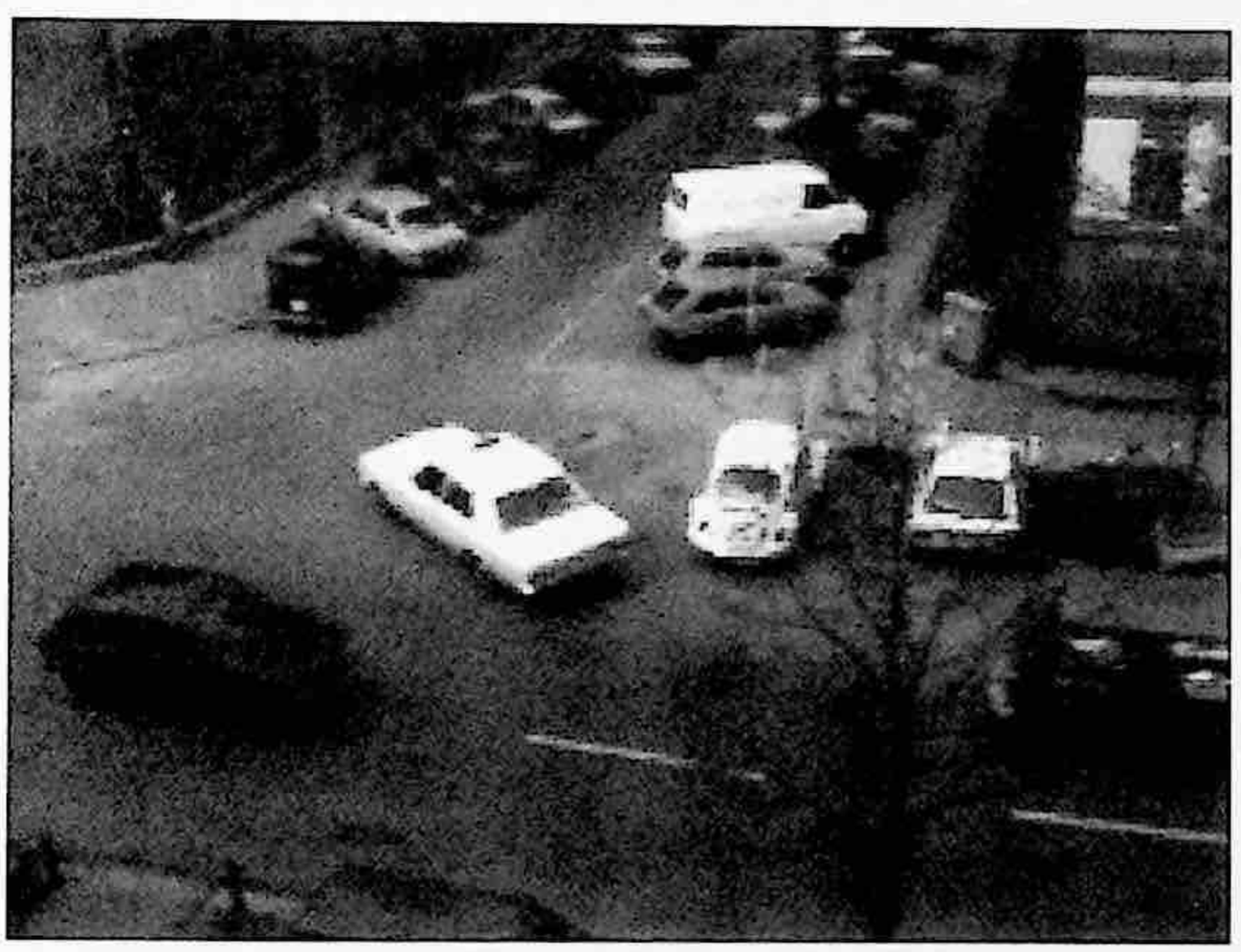


FIGURE 13.16 Hamburg Taxi.

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

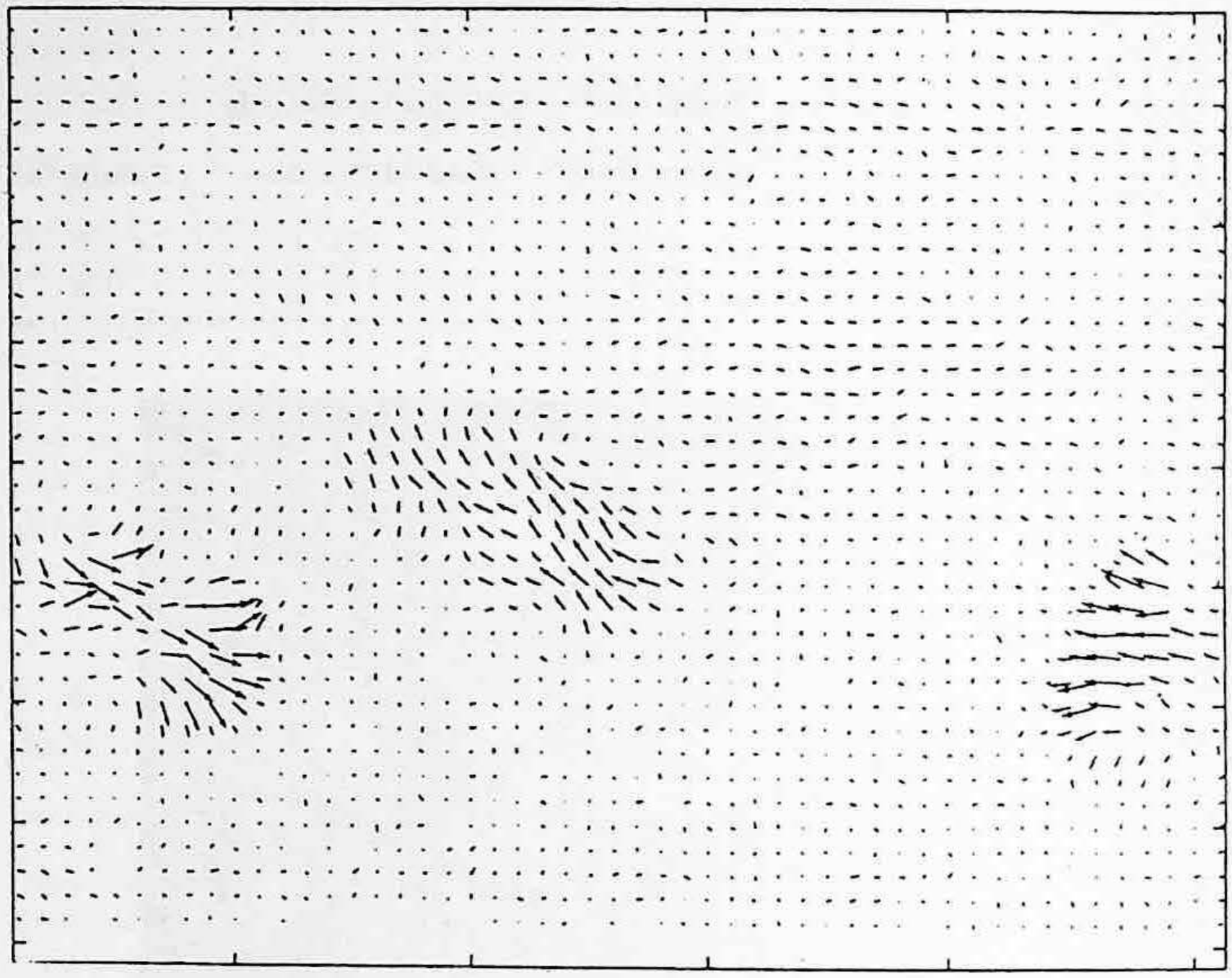
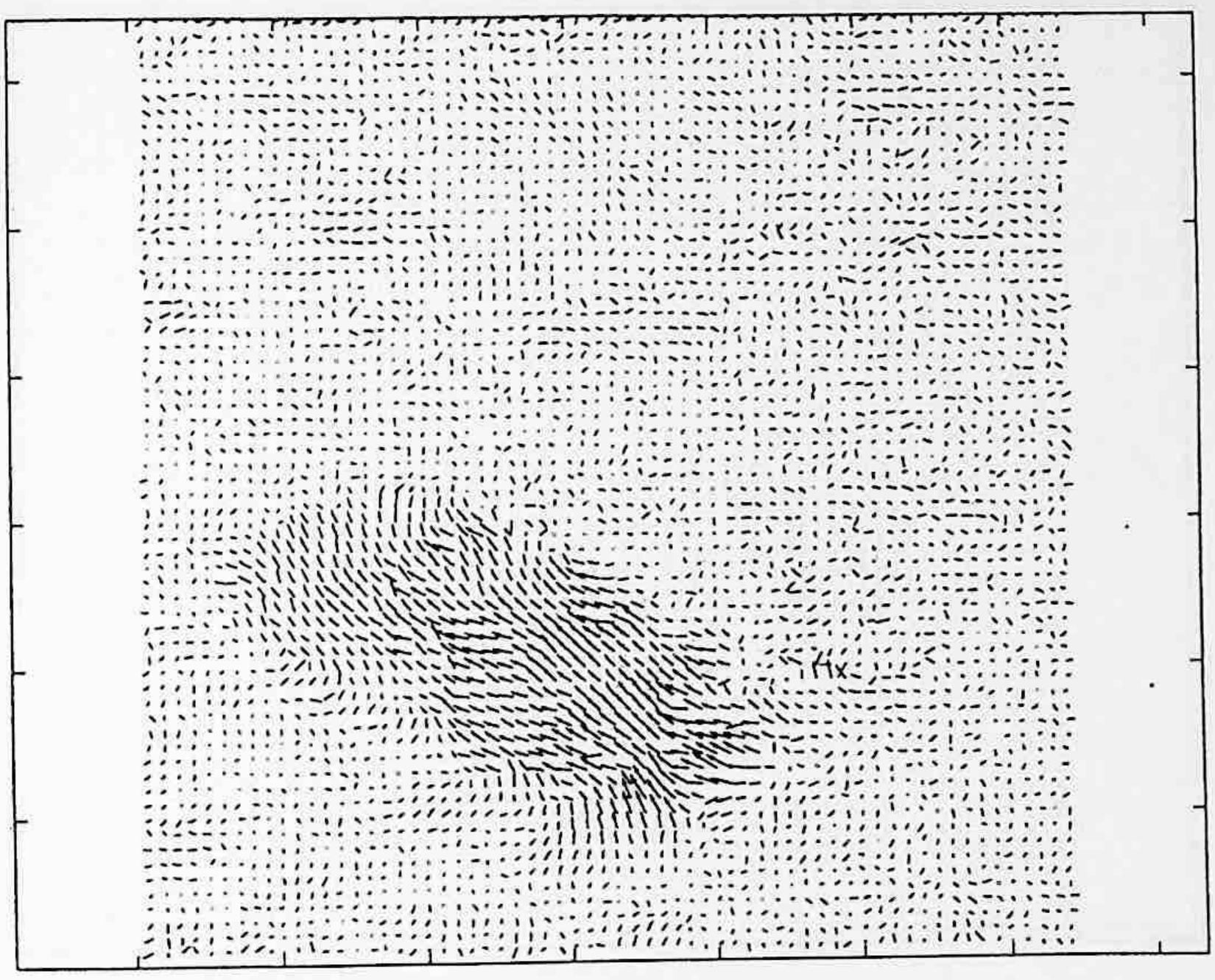


FIGURE 13.17 Needle diagram of flow field of Hamburg Taxi sequence obtained by using the correlation-feedback algorithm.

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.



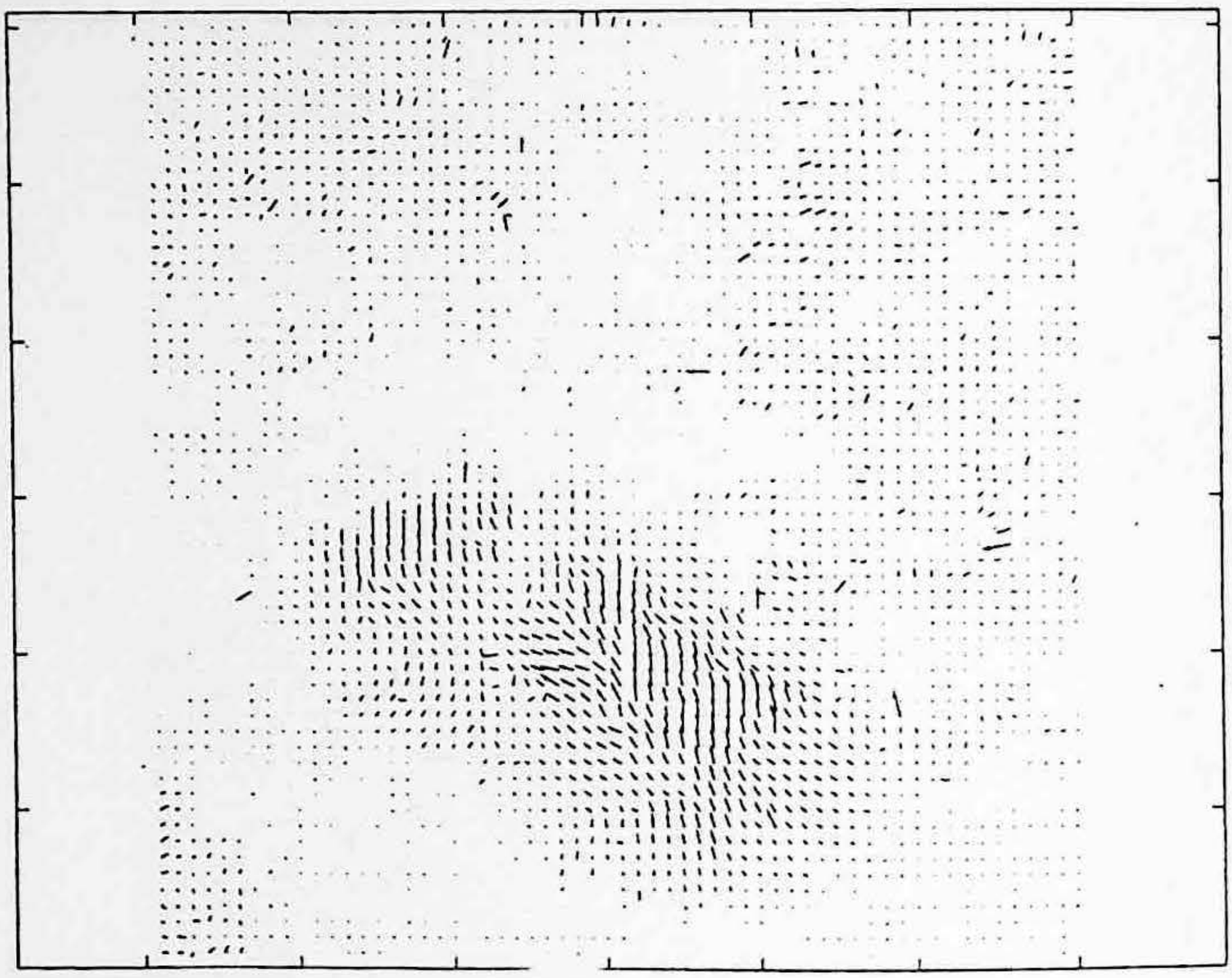
(a)

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)

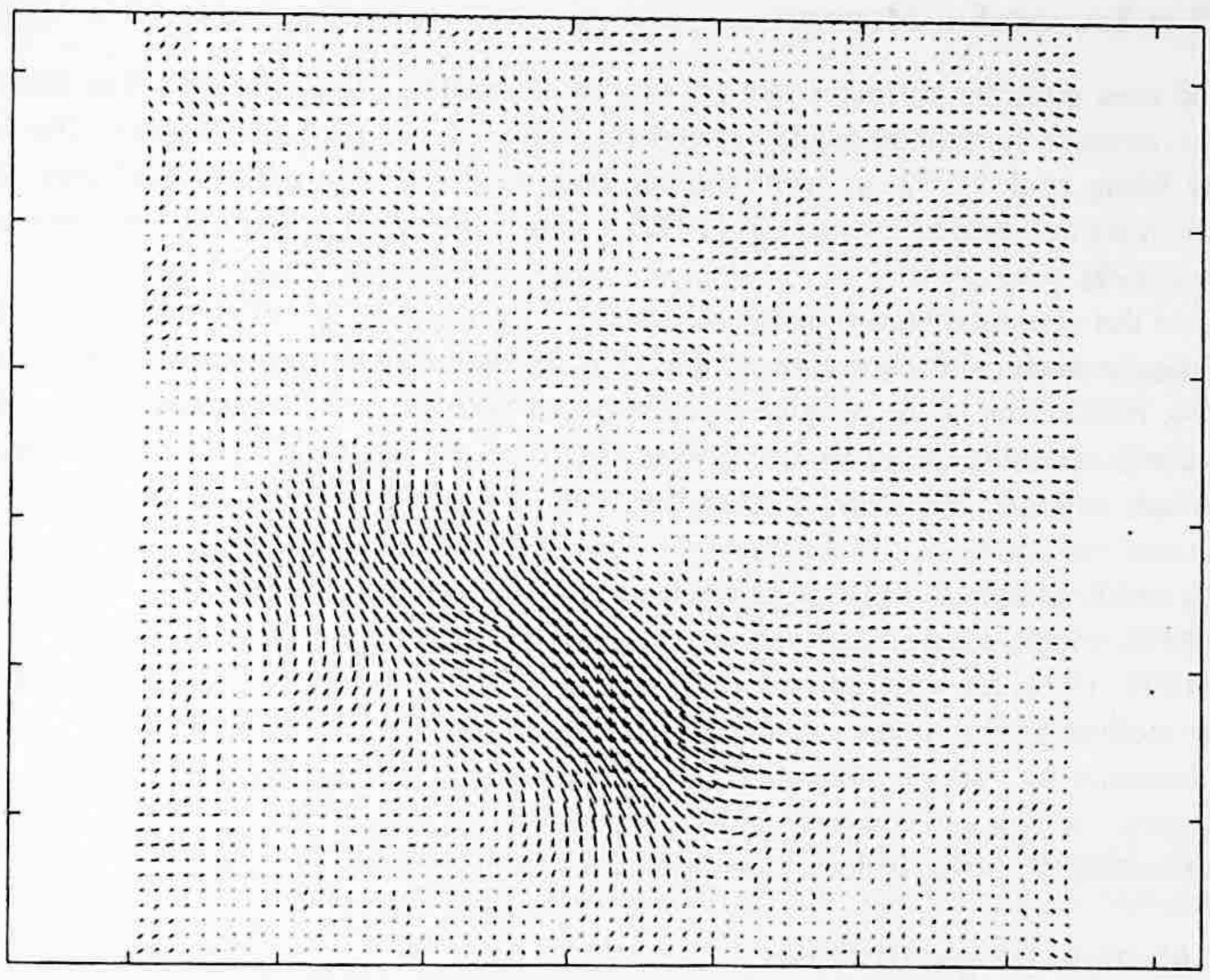
FIGURE 13.18 (continued)

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)

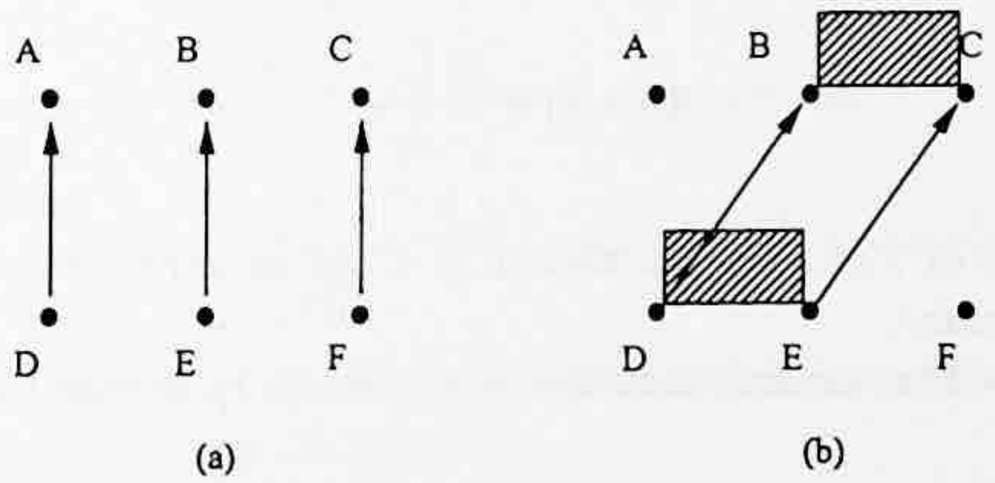


FIGURE 13.19 Multiple attributes vs. single attribute. (a) With intensity information only, points D, E, and F tend to match to points A, B, and C, respectively. (b) With intensity, edge and corner information points D and E tend to match points B and C, respectively.

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_{ij} 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?

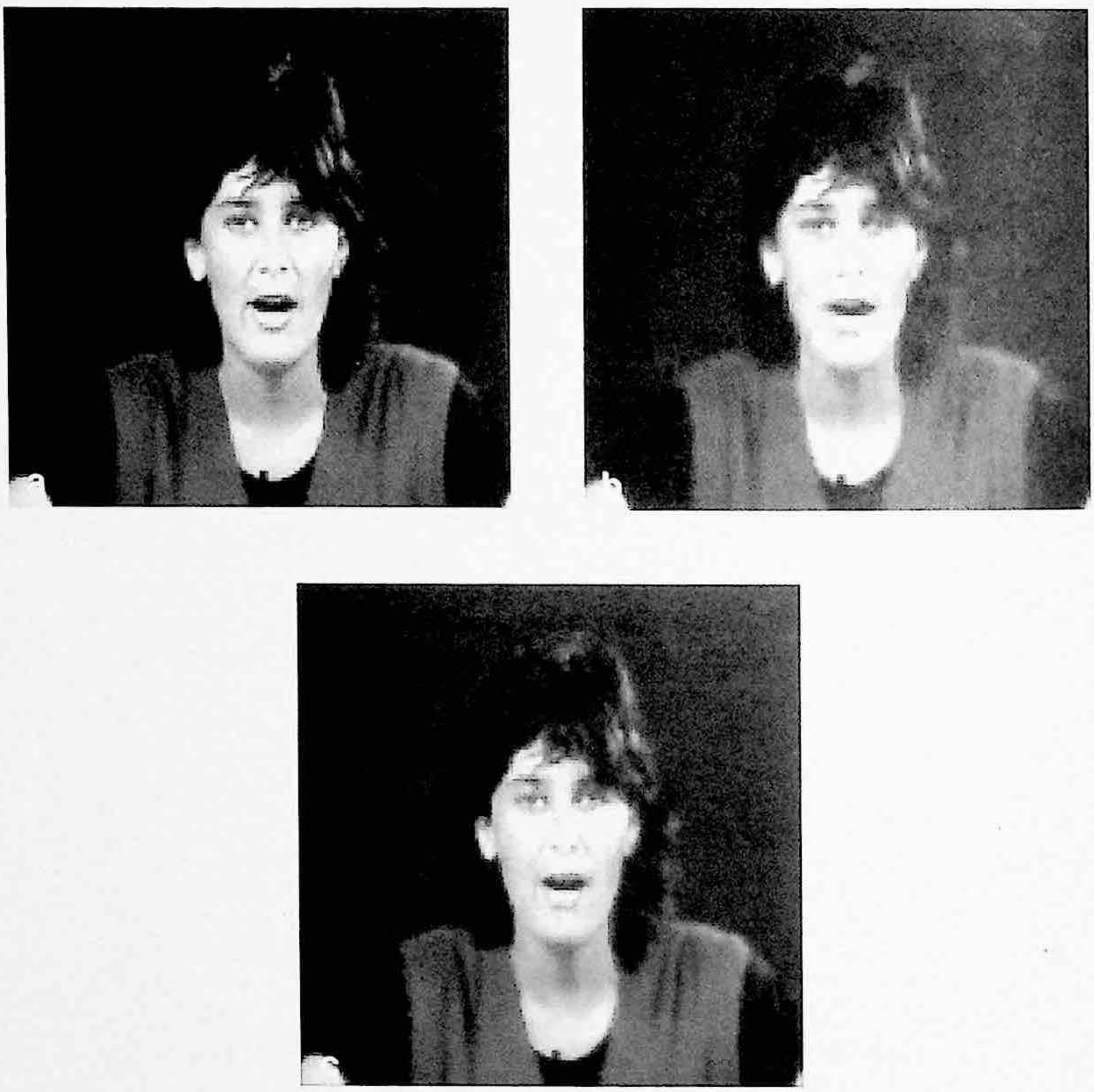


FIGURE 13.20 (a) The 21st original frame of the Miss America sequence; (b) the reconstructed 21st frame with H.263; (c) the reconstructed 21st frame with the proposed algorithm.

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

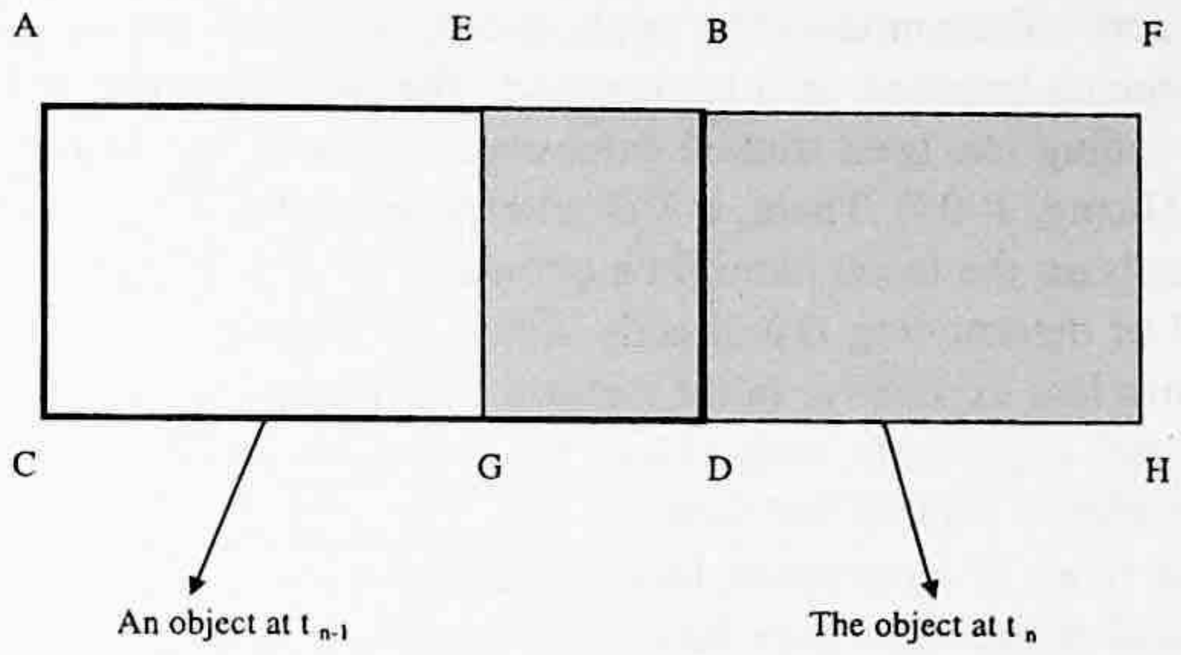


FIGURE 14.1 Occlusion and disocclusion.

(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.

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