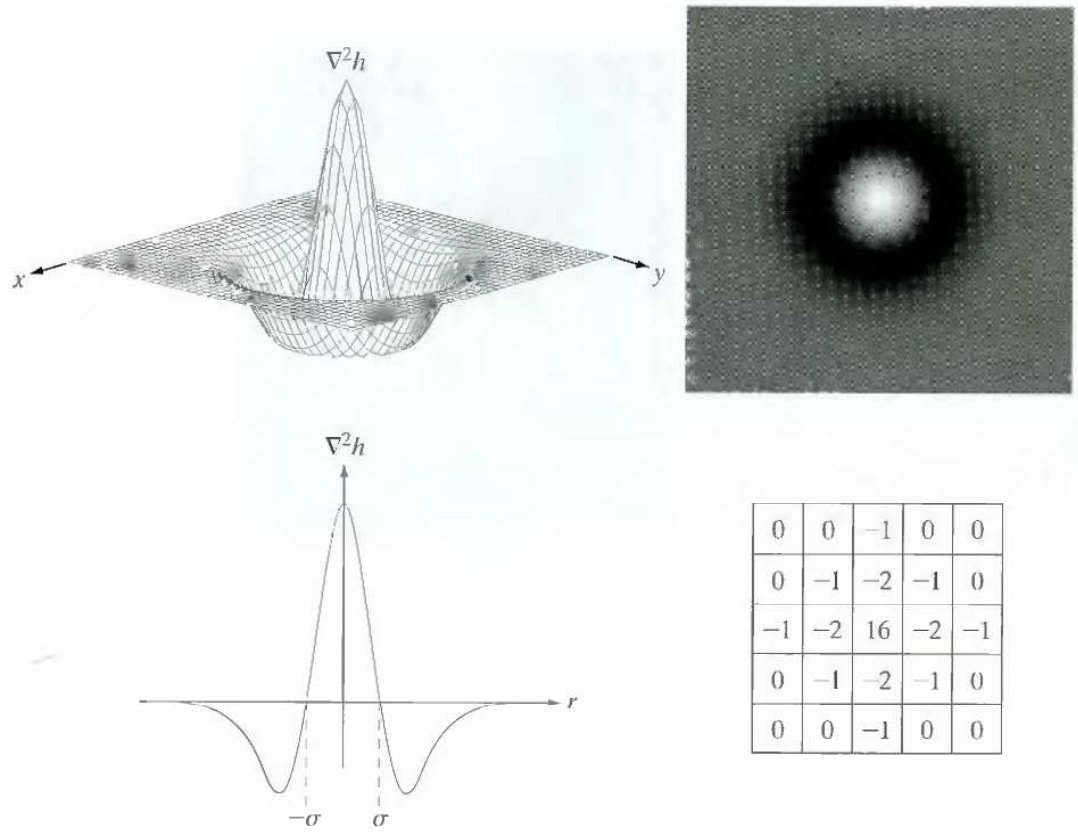


0	0	−1	0	0
0	−1	−2	−1	0
−1	−2	16	−2	−1
0	−1	−2	−1	0
0	0	−1	0	0

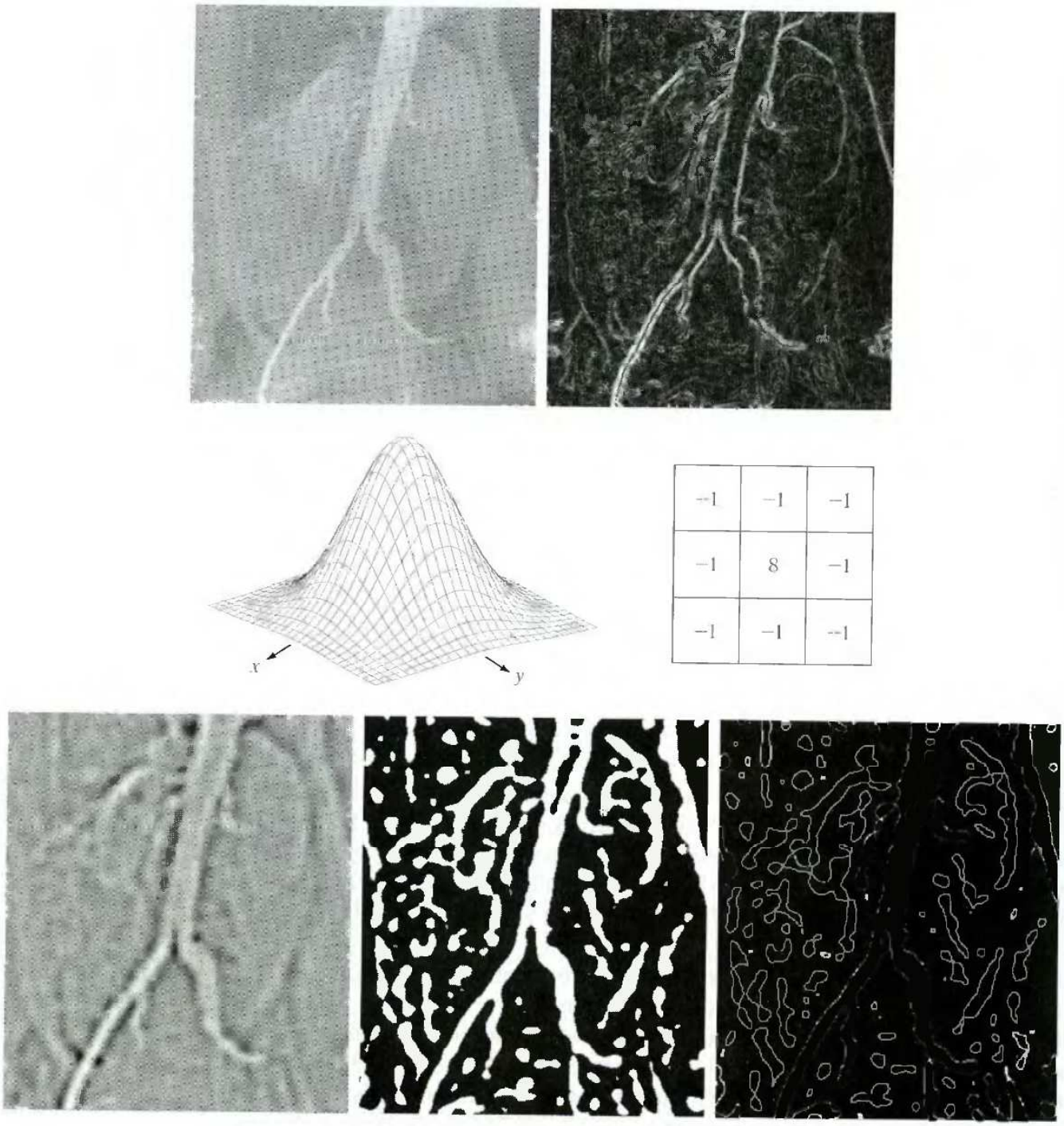
a b
c d

FIGURE 10.14 Laplacian of a Gaussian (LoG). (a) 3-D plot. (b) Image (black is negative, gray is the zero plane, and white is positive). (c) Cross section showing zero crossings. (d) 5 × 5 mask approximation to the shape of (a).

Thus, we see that the purpose of the Gaussian function in the LoG formulation is to smooth the image, and the purpose of the Laplacian operator is to provide an image with zero crossings used to establish the location of edges. Smoothing the image reduces the effect of noise and, in principle, it counters the increased effect of noise caused by the second derivatives of the Laplacian. It is of interest to note that neurophysiological experiments carried out in the early 1980s (Ullman [1981], Marr [1982]) provide evidence that certain aspects of human vision can be modeled mathematically in the basic form of Eq. (10.1-17).

EXAMPLE 10.5: Edge finding by zero crossings.

■ Figure 10.15(a) shows the angiogram image discussed in Section 1.3.2. Figure 10.15(b) shows the Sobel gradient of this image, included here for comparison. Figure 10.15(c) is a spatial Gaussian function (with a standard deviation of five pixels) used to obtain a 27 × 27 spatial smoothing mask. The mask was obtained by sampling this Gaussian function at equal intervals. Figure 10.15(d) is the spatial mask used to implement Eq. (10.1-15). Figure 10.15(e) is the LoG image obtained by smoothing the original image with the Gaussian smoothing mask, followed by application of the Laplacian mask (this image was cropped to eliminate the border effects produced by the smoothing mask). As noted in the preceding paragraph, $\nabla^2 h$ can be computed by application of (c) followed by (d). Employing this procedure provides more control over the smoothing function, and often results in two masks that are much smaller when compared with a single composite mask that implements Eq. (10.1-17) directly. A composite mask usually is larger because it must incorporate the more complex shape shown in Fig. 10.14(a).



a b
c d
e f g

FIGURE 10.15 (a) Original image. (b) Sobel gradient (shown for comparison). (c) Spatial Gaussian smoothing function. (d) Laplacian mask. (e) LoG. (f) Thresholded LoG. (g) Zero crossings. (Original image courtesy of Dr. David R. Pickens, Department of Radiology and Radiological Sciences, Vanderbilt University Medical Center.)

The LoG result shown in Fig. 10.15(e) is the image from which zero crossings are computed to find edges. One straightforward approach for approximating zero crossings is to threshold the LoG image by setting all its positive values to, say, white, and all negative values to black. The result is shown in Fig. 10.15(f). The logic behind this approach is that zero crossings occur between positive and negative values of the Laplacian. Finally, Fig. 10.15(g) shows the estimated zero crossings, obtained by scanning the thresholded image and noting the transitions between black and white.

Comparing Figs. 10.15(b) and (g) reveals several interesting and important differences. First, we note that the edges in the zero-crossing image are thinner than the gradient edges. This is a characteristic of zero crossings that makes this approach attractive. On the other hand, we see in Fig. 10.15(g) that the edges determined by zero crossings form numerous closed loops. This so-called spaghetti effect is one of the most serious drawbacks of this method. Another major drawback is the computation of zero crossings, which is the foundation of the method. Although it was reasonably straightforward in this example, the computation of zero crossings presents a challenge in general, and considerably more sophisticated techniques often are required to obtain acceptable results (Huertas and Medione [1986]).

Zero-crossing methods are of interest because of their noise reduction capabilities and potential for rugged performance. However, the limitations just noted present a significant barrier in practical applications. For this reason, edge-finding techniques based on various implementations of the gradient still are used more frequently than zero crossings in the implementation of segmentation algorithms. ■

10.2 Edge Linking and Boundary Detection

Ideally, the methods discussed in the previous section should yield pixels lying only on edges. In practice, this set of pixels seldom characterizes an edge completely because of noise, breaks in the edge from nonuniform illumination, and other effects that introduce spurious intensity discontinuities. Thus edge detection algorithms typically are followed by linking procedures to assemble edge pixels into meaningful edges. Several basic approaches are suited to this purpose.

10.2.1 Local Processing

One of the simplest approaches for linking edge points is to analyze the characteristics of pixels in a small neighborhood (say, 3×3 or 5×5) about every point (x, y) in an image that has been labeled an edge point by one of the techniques discussed in the previous section. All points that are similar according to a set of predefined criteria are linked, forming an edge of pixels that share those criteria.

The two principal properties used for establishing similarity of edge pixels in this kind of analysis are (1) the strength of the response of the gradient operator used to produce the edge pixel; and (2) the direction of the gradient vector. The first property is given by the value of ∇f, as defined in Eq. (10.1-4) or (10.1-12). Thus an edge pixel with coordinates (x_0, y_0) in a predefined neighborhood of

(x, y), is similar in magnitude to the pixel at (x, y) if

$$|\nabla f(x, y) - \nabla f(x_0, y_0)| \leq E \tag{10.2-1}$$

where E is a nonnegative threshold.

The direction (angle) of the gradient vector is given by Eq. (10.1-5). An edge pixel at (x_0, y_0) in the predefined neighborhood of (x, y) has an angle similar to the pixel at (x, y) if

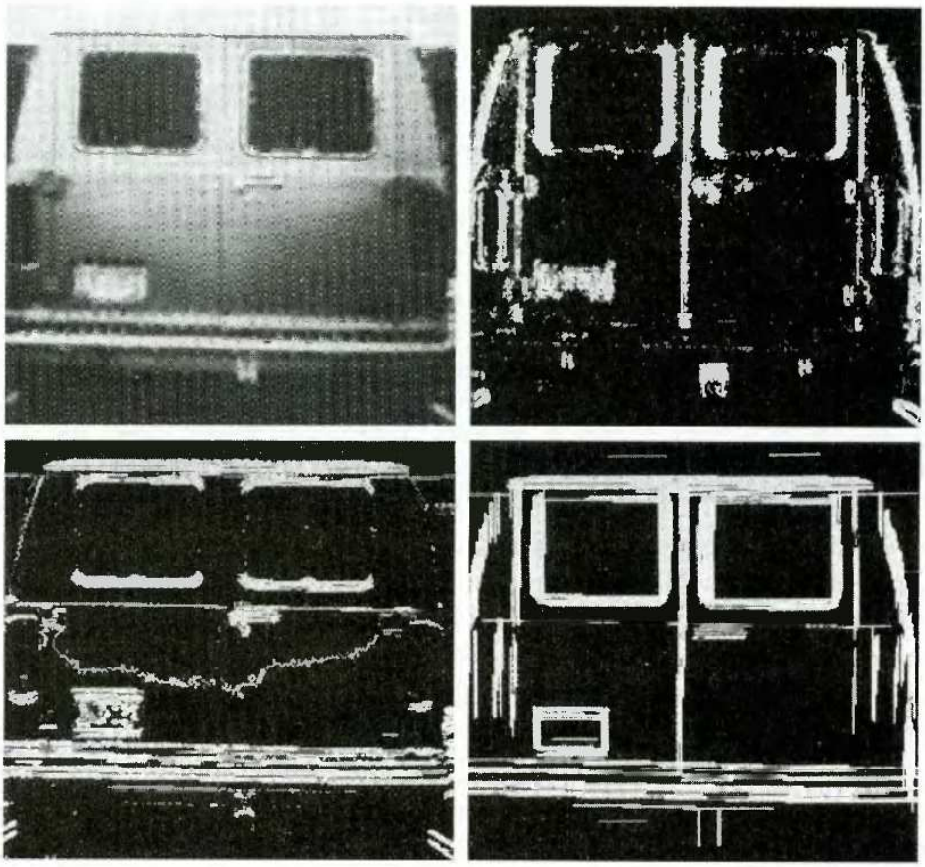
$$|\alpha(x, y) - \alpha(x_0, y_0)| < A \tag{10.2-2}$$

where A is a nonnegative angle threshold. As noted in Eq. (10.1-5), the direction of the edge at (x, y) is *perpendicular* to the direction of the gradient vector at that point.

A point in the predefined neighborhood of (x, y) is linked to the pixel at (x, y) if both magnitude and direction criteria are satisfied. This process is repeated at every location in the image. A record must be kept of linked points as the center of the neighborhood is moved from pixel to pixel. A simple bookkeeping procedure is to assign a different gray level to each set of linked edge pixels.

EXAMPLE 10.6: Edge-point linking based on local processing.

■ To illustrate the foregoing procedure, consider Fig. 10.16(a), which shows an image of the rear of a vehicle. The objective is to find rectangles whose sizes makes them suitable candidates for license plates. The formation of these rectangles can be accomplished by detecting strong horizontal and vertical edges. Figures 10.16(b) and (c) show vertical and horizontal edges obtained by using



a b
c d

FIGURE 10.16 (a) Input image. (b) G_y component of the gradient. (c) G_x component of the gradient. (d) Result of edge linking. (Courtesy of Perceptics Corporation.)

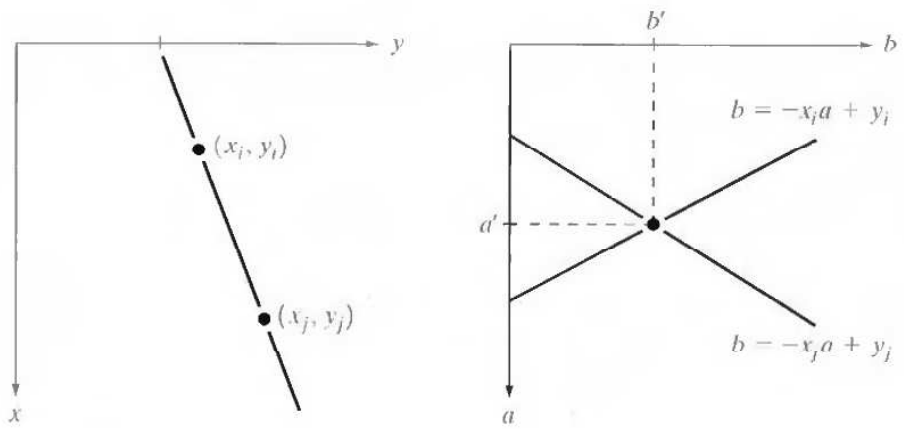
the horizontal and vertical Sobel operators. Figure 10.16(d) shows the result of linking all points that simultaneously had a gradient value greater than 25 and whose gradient directions did not differ by more than 15°. The horizontal lines were formed by sequentially applying these criteria to every row of Fig. 10.16(c). A sequential column scan of Fig. 10.16(b) yielded the vertical lines. Further processing consisted of linking edge segments separated by small breaks and deleting isolated short segments. As Fig. 10.16(d) shows, the rectangle corresponding to the license plate was one of the few rectangles detected in the image. It would be a simple matter to locate the license plate based on these rectangles (the width-to-height ratio of the license plate rectangle has a distinctive 2:1 proportion for U.S. plates). ■

10.2.2 Global Processing via the Hough Transform

In this section, points are linked by determining first if they lie on a curve of specified shape. Unlike the local analysis method discussed in Section 10.2.1, we now consider global relationships between pixels.

Given n points in an image, suppose that we want to find subsets of these points that lie on straight lines. One possible solution is to first find all lines determined by every pair of points and then find all subsets of points that are close to particular lines. The problem with this procedure is that it involves finding $n(n-1)/2 \sim n^2$ lines and then performing $(n)(n(n-1))/2 \sim n^3$ comparisons of every point to all lines. This approach is computationally prohibitive in all but the most trivial applications.

Hough [1962] proposed an alternative approach, commonly referred to as the *Hough transform*. Consider a point (x_i, y_i) and the general equation of a straight line in slope-intercept form, $y_i = ax_i + b$. Infinitely many lines pass through (x_i, y_i), but they all satisfy the equation $y_i = ax_i + b$ for varying values of a and b. However, writing this equation as $b = -x_i a + y_i$ and considering the ab-plane (also called *parameter space*) yields the equation of a *single* line for a fixed pair (x_i, y_i). Furthermore, a second point (x_j, y_j) also has a line in parameter space associated with it, and this line intersects the line associated with (x_i, y_i) at (a', b'), where a' is the slope and b' the intercept of the line containing both (x_i, y_i) and (x_j, y_j) in the xy-plane. In fact, all points contained on this line have lines in parameter space that intersect at (a', b'). Figure 10.17 illustrates these concepts.



a b

FIGURE 10.17 (a) xy-plane. (b) Parameter space.

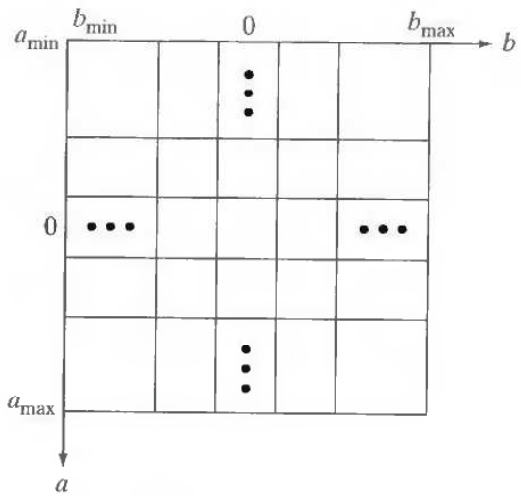


FIGURE 10.18 Subdivision of the parameter plane for use in the Hough transform.

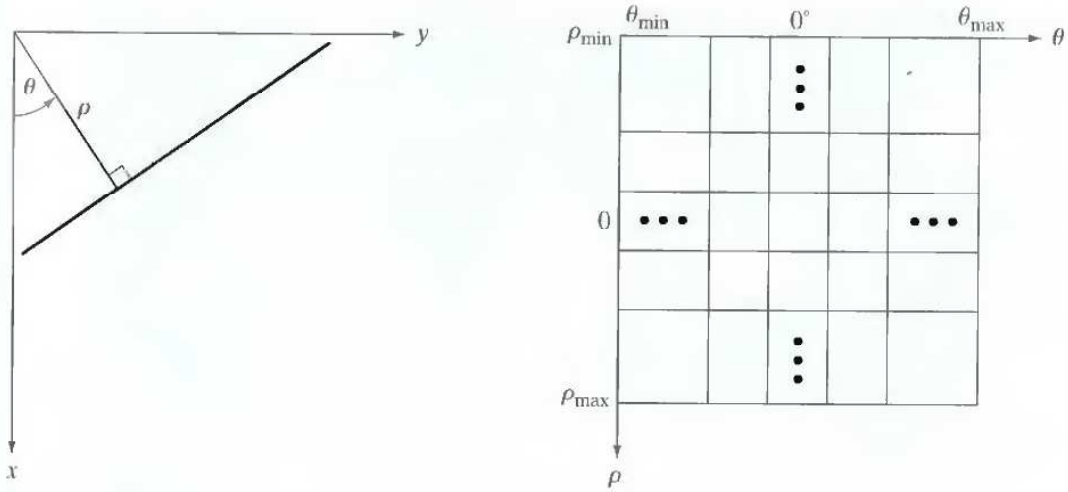
The computational attractiveness of the Hough transform arises from subdividing the parameter space into so-called *accumulator cells*, as illustrated in Fig. 10.18, where (a_{max}, a_{min}) and (b_{max}, b_{min}) are the expected ranges of slope and intercept values. The cell at coordinates (i, j), with accumulator value $A(i, j)$, corresponds to the square associated with parameter space coordinates (a_i, b_j). Initially, these cells are set to zero. Then, for every point (x_k, y_k) in the image plane, we let the parameter a equal each of the allowed subdivision values on the a-axis and solve for the corresponding b using the equation $b = -x_k a + y_k$. The resulting b's are then rounded off to the nearest allowed value in the b-axis. If a choice of a_p results in solution b_q, we let $A(p, q) = A(p, q) + 1$. At the end of this procedure, a value of Q in $A(i, j)$ corresponds to Q points in the xy-plane lying on the line $y = a_i x + b_j$. The number of subdivisions in the ab-plane determines the accuracy of the colinearity of these points.

Note that subdividing the a axis into K increments gives, for every point (x_k, y_k), K values of b corresponding to the K possible values of a. With n image points, this method involves nK computations. Thus the procedure just discussed is *linear* in n, and the product nK does not approach the number of computations discussed at the beginning of this section unless K approaches or exceeds n.

A problem with using the equation $y = ax + b$ to represent a line is that the slope approaches infinity as the line approaches the vertical. One way around this difficulty is to use the normal representation of a line:

$$x \cos\theta + y \sin\theta = \rho. \tag{10.2-3}$$

Figure 10.19(a) illustrates the geometrical interpretation of the parameters used in Eq. (10.2-3). The use of this representation in constructing a table of accumulators is identical to the method discussed for the slope-intercept representation. Instead of straight lines, however, the loci are sinusoidal curves in the $\rho\theta$-plane. As before, Q collinear points lying on a line $x \cos\theta_j + y \sin\theta_j = \rho_i$ yield Q sinusoidal curves that intersect at (ρ_i, θ_j) in the parameter space. Incrementing θ and solving for the corresponding ρ gives Q entries in accumulator $A(i, j)$ associated with the cell determined by (ρ_i, θ_j). Figure 10.19(b) illustrates the subdivision of the parameter space.



a b

FIGURE 10.19 (a) Normal representation of a line. (b) Subdivision of the $\rho\theta$-plane into cells.

The range of angle θ is $\pm 90°$, measured with respect to the x-axis. Thus with reference to Fig. 10.19(a), a horizontal line has $\theta = 0°$, with ρ being equal to the positive x-intercept. Similarly, a vertical line has $\theta = 90°$, with ρ being equal to the positive y-intercept, or $\theta = -90°$, with ρ being equal to the negative y-intercept.

EXAMPLE 10.7: Illustration of the Hough transform.

■ Figure 10.20 illustrates the Hough transform based on Eq. (10.2-3). Figure 10.20(a) shows an image with five labeled points. Each of these points is mapped onto the $\rho\theta$-plane, as shown in Fig. 10.20(b). The range of θ values is $\pm 90°$, and the range of the ρ-axis is $\pm\sqrt{2}D$, where D is the distance between corners in the image. Unlike the transform based on using the slope intercept, each of these curves has a different sinusoidal shape. The horizontal line resulting from the mapping of point 1 is a special case of a sinusoid with zero amplitude.

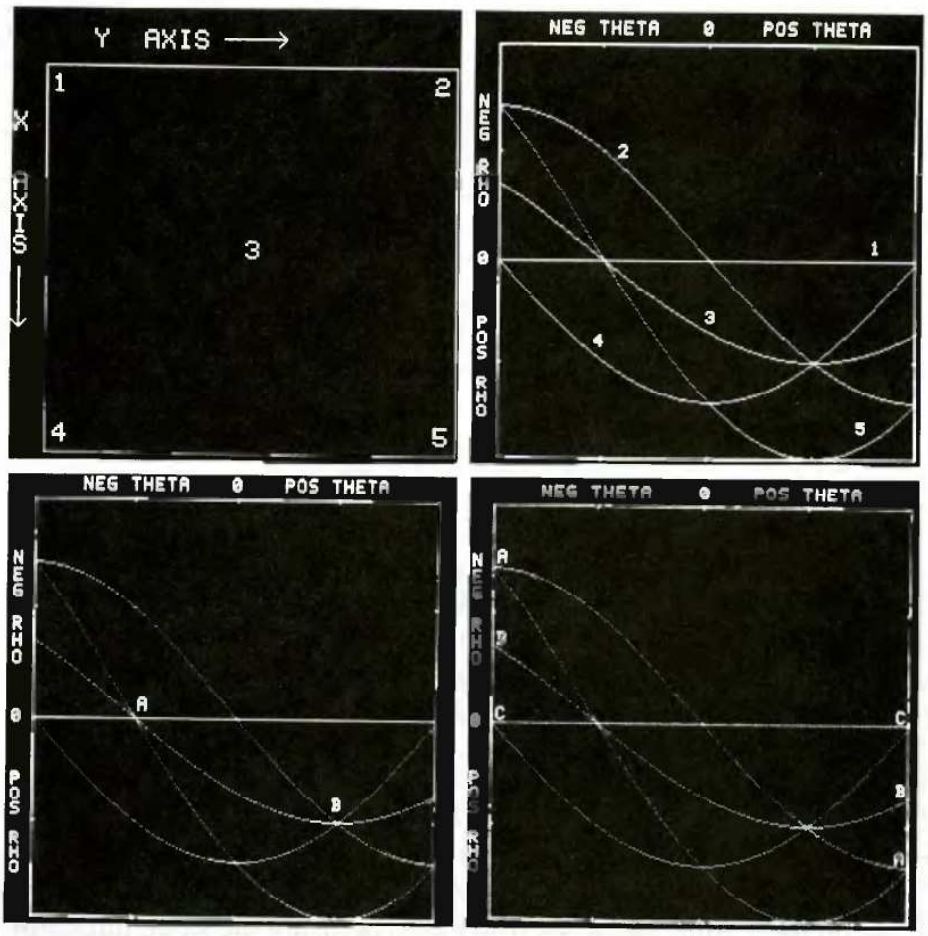
The colinearity detection property of the Hough transform is illustrated in Fig. 10.20(c). Point A (not to be confused with accumulator values) denotes the intersection of the curves corresponding to points 1, 3, and 5 in the xy-image plane. The location of point A indicates that these three points lie on a straight line passing through the origin ($\rho = 0$) and oriented at $-45°$. Similarly, the curves intersecting at point B in the parameter space indicate that points 2, 3, and 4 lie on a straight line oriented at 45° and whose distance from the origin is one-half the diagonal distance from the origin of the image to the opposite corner.

Finally, Fig. 10.20(d) indicates the fact that the Hough transform exhibits a reflective adjacency relationship at the right and left edges of the parameter space. This property, shown by the points marked A, B, and C in Fig. 10.20(d), is the result of the manner in which θ and ρ change sign at the $\pm 90°$ boundaries. ■

Although the focus so far has been on straight lines, the Hough transform is applicable to any function of the form $g(\mathbf{v}, \mathbf{c}) = 0$, where $\mathbf{v}$ is a vector of coordinates and $\mathbf{c}$ is a vector of coefficients. For example, the points lying on the circle

$$(x - c_1)^2 + (y - c_2)^2 = c_3^2 \tag{10.2-4}$$

can be detected by using the approach just discussed. The basic difference is the presence of three parameters (c_1, c_2, and c_3), which results in a 3-D parameter



a b
c d

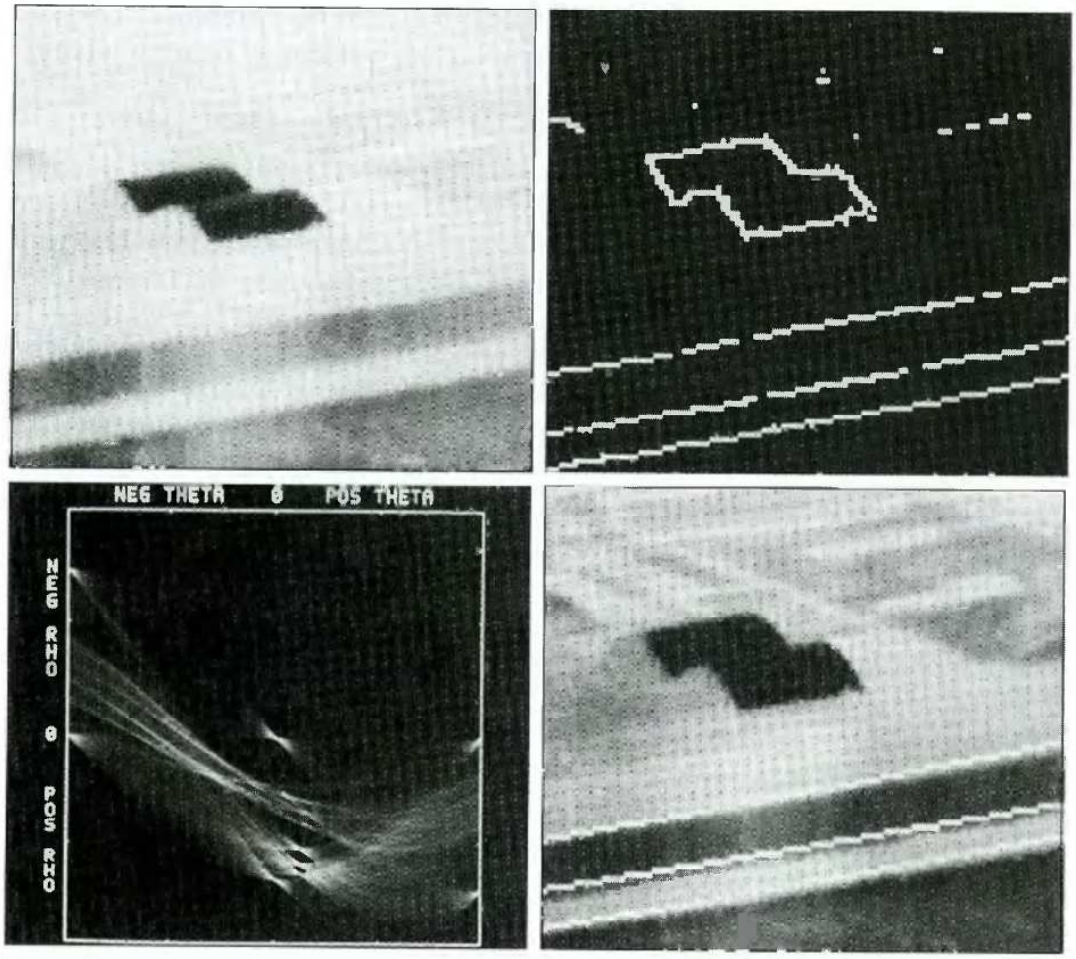
FIGURE 10.20 Illustration of the Hough transform. (Courtesy of Mr. D. R. Cate, Texas Instruments, Inc.)

space with cubelike cells and accumulators of the form $A(i, j, k)$. The procedure is to increment c_1 and c_2, solve for the c_3 that satisfies Eq. (10.2-4), and update the accumulator corresponding to the cell associated with the triplet (c_1, c_2, c_3). Clearly, the complexity of the Hough transform is proportional to the number of coordinates and coefficients in a given functional representation. Further generalizations of the Hough transform to detect curves with no simple analytic representations are possible, as is the application of the transform to gray-scale images. Several references dealing with these extensions are included at the end of this chapter.

We now return to the edge-linking problem. An approach based on the Hough transform is as follows:

1. Compute the gradient of an image and threshold it to obtain a binary image.
2. Specify subdivisions in the $\rho\theta$-plane.
3. Examine the counts of the accumulator cells for high pixel concentrations.
4. Examine the relationship (principally for continuity) between pixels in a chosen cell.

The concept of continuity in this case usually is based on computing the distance between disconnected pixels identified during traversal of the set of pixels corresponding to a given accumulator cell. A gap at any point is significant if the distance



a b
c d

FIGURE 10.21 (a) Infrared image. (b) Thresholded gradient image. (c) Hough transform. (d) Linked pixels. (Courtesy of Mr. D. R. Cate, Texas Instruments, Inc.)

between that point and its closest neighbor exceeds a certain threshold. (See Section 2.5 for a discussion of connectivity, neighborhoods, and distance measures.)

EAMPLE 10.8: Using the Hough transform for edge linking.

■ Figure 10.21(a) shows an aerial infrared image containing two hangars and a runway. Figure 10.21(b) is a thresholded gradient image obtained using the Sobel operators discussed in Section 10.1.3 (note the small gaps in the borders of the runway). Figure 10.21(c) shows the Hough transform of the gradient image. Figure 10.21(d) shows (in white) the set of pixels linked according to the criteria that (1) they belonged to one of the three accumulator cells with the highest count, and (2) no gaps were longer than five pixels. Note the disappearance of the gaps as a result of linking. ■

10.2.3 Global Processing via Graph-Theoretic Techniques

In this section we discuss a global approach for edge detection and linking based on representing edge segments in the form of a graph and searching the graph for low-cost paths that correspond to significant edges. This representation provides a rugged approach that performs well in the presence of noise. As might be expected, the procedure is considerably more complicated and requires more processing time than the methods discussed so far.

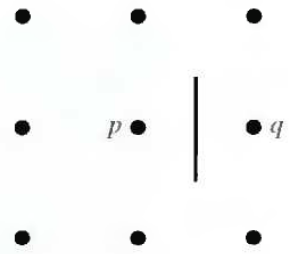


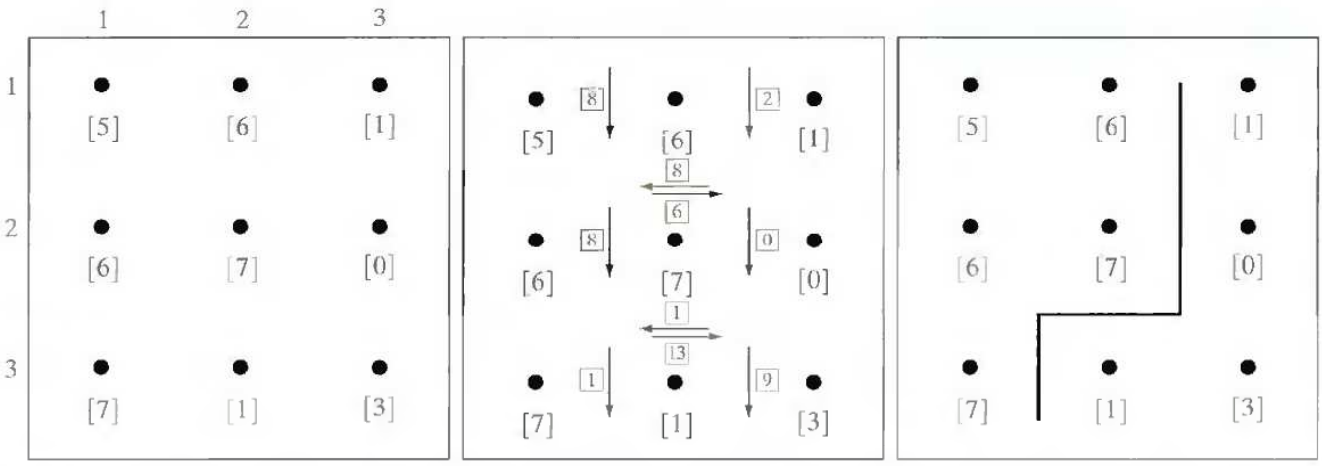
FIGURE 10.22 Edge element between pixels p and q.

We begin the development with some basic definitions. A *graph* $G = (N, U)$ is a finite, nonempty set of nodes N, together with a set U of unordered pairs of distinct elements of N. Each pair (n_i, n_j) of U is called an *arc*. A graph in which the arcs are directed is called a *directed graph*. If an arc is directed from node n_i to node n_j, then n_j is said to be a *successor* of the *parent* node n_i. The process of identifying the successors of a node is called *expansion* of the node. In each graph we define levels, such that level 0 consists of a single node, called the *start* or *root* node, and the nodes in the last level are called *goal* nodes. A *cost* $c(n_i, n_j)$ can be associated with every arc (n_i, n_j). A sequence of nodes $n_1, n_2, \ldots, n_k$, with each node n_i being a successor of node n_{i-1}, is called a *path* from n_1 to n_k. The cost of the entire path is

$$c = \sum_{i=2}^{k} c(n_{i-1}, n_i). \tag{10.2-5}$$

The following discussion is simplified if we define an *edge element* as the boundary between two pixels p and q, such that p and q are 4-neighbors, as Fig. 10.22 illustrates. Edge elements are identified by the xy-coordinates of points p and q. In other words, the edge element in Fig. 10.22 is defined by the pairs $(x_p, y_p)(x_q, y_q)$. Consistent with the definition given in Section 10.1.3, an *edge* is a sequence of connected edge elements.

We can illustrate how the concepts just discussed apply to edge detection using the 3×3 image shown in Fig. 10.23(a). The outer numbers are pixel



a b c

FIGURE 10.23 (a) A 3×3 image region. (b) Edge segments and their costs. (c) Edge corresponding to the lowest-cost path in the graph shown in Fig. 10.24.

coordinates and the numbers in brackets represent gray-level values. Each edge element, defined by pixels p and q, has an associated cost, defined as

$$c(p, q) = H - [f(p) - f(q)] \tag{10.2-6}$$

where H is the highest gray-level value in the image (7 in this case), and $f(p)$ and $f(q)$ are the gray-level values of p and q, respectively. By convention, the point p is on the right-hand side of the direction of travel along edge elements. For example, the edge segment $(1, 2)(2, 2)$ is between points $(1, 2)$ and $(2, 2)$ in Fig. 10.23(b). If the direction of travel is to the right, then p is the point with coordinates $(2, 2)$ and q is point with coordinates $(1, 2)$; therefore, $c(p, q) = 7 - [7 - 6] = 6$. This cost is shown in the box below the edge segment. If, on the other hand, we are traveling to the *left* between the same two points, then p is point $(1, 2)$ and q is $(2, 2)$. In this case the cost is 8, as shown above the edge segment in Fig. 10.23(b). To simplify the discussion, we assume that edges start in the top row and terminate in the last row, so that the first element of an edge can be only between points $(1, 1)$, $(1, 2)$ or $(1, 2)$, $(1, 3)$. Similarly, the last edge element has to be between points $(3, 1)$, $(3, 2)$ or $(3, 2)$, $(3, 3)$. Keep in mind that p and q are 4-neighbors, as noted earlier.

Figure 10.24 shows the graph for this problem. Each node (rectangle) in the graph corresponds to an edge element from Fig. 10.23. An arc exists between two nodes if the two corresponding edge elements taken in succession can be part

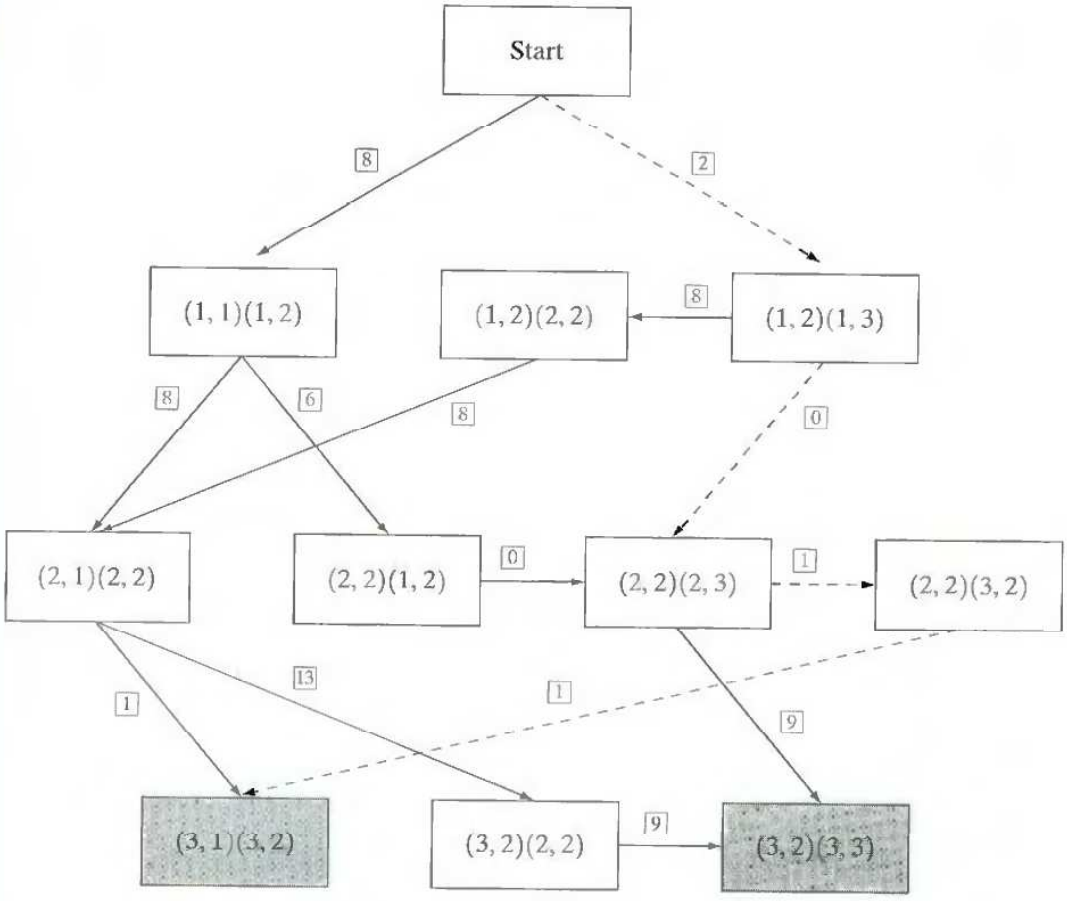


FIGURE 10.24 Graph for the image in Fig. 10.23(a). The lowest-cost path is shown dashed.

of an edge. As in Fig. 10.23(b), the cost of each edge segment, computed using Eq. (10.2-6), is shown in a box on the side of the arc leading into the corresponding node. Goal nodes are shown shaded. The minimum cost path is shown dashed, and the edge corresponding to this path is shown in Fig. 10.23(c).

In general, the problem of finding a minimum-cost path is not trivial in terms of computation. Typically, the approach is to sacrifice optimality for the sake of speed, and the following algorithm represents a class of procedures that use heuristics in order to reduce the search effort. Let $r(n)$ be an estimate of the cost of a minimum-cost path from the start node s to a goal node, where the path is constrained to go through n. This cost can be expressed as the estimate of the cost of a minimum-cost path from s to n plus an estimate of the cost of that path from n to a goal node; that is,

$$r(n) = g(n) + h(n). \tag{10.2-7}$$

Here, $g(n)$ can be chosen as the lowest-cost path from s to n found so far, and $h(n)$ is obtained by using any available heuristic information (such as expanding only certain nodes based on previous costs in getting to that node). An algorithm that uses $r(n)$ as the basis for performing a graph search is as follows:

Step 1: Mark the start node OPEN and set $g(s) = 0$.
Step 2: If no node is OPEN exit with failure; otherwise, continue.
Step 3: Mark CLOSED the OPEN node n whose estimate $r(n)$ computed from Eq. (10.2-7) is smallest. (Ties for minimum r values are resolved arbitrarily, but always in favor of a goal node.)
Step 4: If n is a goal node, exit with the solution path obtained by tracing back through the pointers; otherwise, continue.
Step 5: Expand node n, generating all of its successors. (If there are no successors go to step 2.)
Step 6: If a successor n_i is not marked, set

$$r(n_i) = g(n) + c(n, n_i),$$

mark it OPEN, and direct pointers from it back to n.
Step 7: If a successor n_i is marked CLOSED or OPEN, update its value by letting

$$g'(n_i) = \min[g(n_i), g(n) + c(n, n_i)].$$

Mark OPEN those CLOSED successors whose g' values were thus lowered and redirect to n the pointers from all nodes whose g' values were lowered. Go to step 2.

This algorithm does not guarantee a minimum-cost path; its advantage is speed via the use of heuristics. However, if $h(n)$ is a lower bound on the cost of the minimal-cost path from node n to a goal node, the procedure indeed yields an optimal path to a goal (Hart et al. [1968]). If no heuristic information is available (that is, $h = 0$), the procedure reduces to the *uniform-cost algorithm* of Dijkstra [1959].

EXAMPLE 10.9: Edge finding by graph search.

■ Figure 10.25 shows an image of a noisy chromosome silhouette and an edge found using a heuristic graph search based on the algorithm developed in this

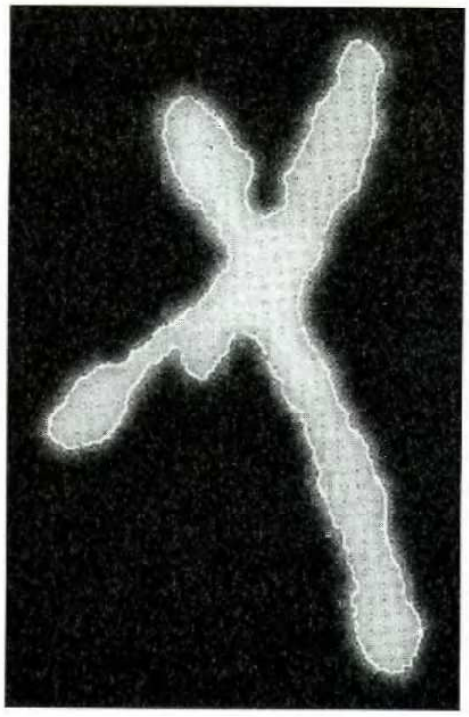


FIGURE 10.25 Image of noisy chromosome silhouette and edge boundary (in white) determined by graph search.

section. The edge is shown in white, superimposed on the original image. Note that in this case the edge and the boundary of the object are approximately the same. The cost was based on Eq. (10.2-6), and the heuristic used at any point on the graph was to determine and use the optimum path for five levels down from that point. Considering the amount of noise present in this image, the graph-search approach yielded a reasonably accurate result. ■

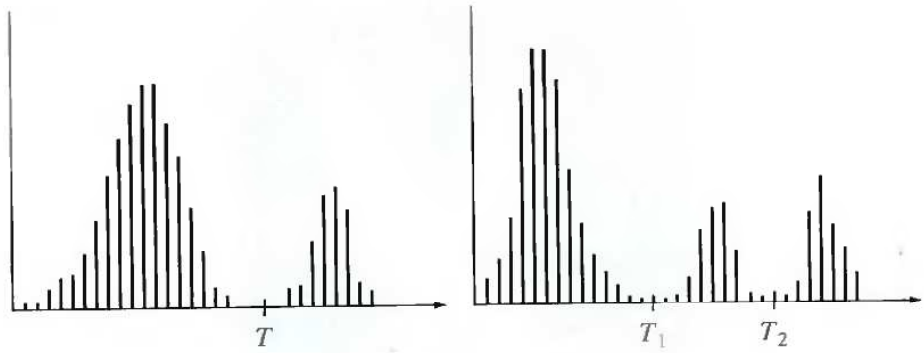
10.3 Thresholding

Because of its intuitive properties and simplicity of implementation, image thresholding enjoys a central position in applications of image segmentation. Simple thresholding was first introduced in Section 3.1, and we have used it in various discussions in the preceding chapters. In this section, we introduce thresholding in a more formal way and extend it to techniques that are considerably more general than what has been presented thus far.

10.3.1 Foundation

Suppose that the gray-level histogram shown in Fig. 10.26(a) corresponds to an image, $f(x, y)$, composed of light objects on a dark background, in such a way that object and background pixels have gray levels grouped into two dominant modes. One obvious way to extract the objects from the background is to select a threshold T that separates these modes. Then any point (x, y) for which $f(x, y) > T$ is called an *object point*; otherwise, the point is called a *background point*. This is the type of thresholding introduced in Section 3.1.

Figure 10.26(b) shows a slightly more general case of this approach, where three dominant modes characterize the image histogram (for example, two types



a b

FIGURE 10.26 (a) Gray-level histograms that can be partitioned by (a) a single threshold, and (b) multiple thresholds.

of light objects on a dark background). Here, *multilevel thresholding* classifies a point (x, y) as belonging to one object class if $T_1 < (x, y) \leq T_2$, to the other object class if $f(x, y) > T_2$, and to the background if $f(x, y) \leq T_1$. In general, segmentation problems requiring multiple thresholds are best solved using region growing methods, such as those discussed in Section 10.4.

Based on the preceding discussion, thresholding may be viewed as an operation that involves tests against a function T of the form

$$T = T[x, y, p(x, y), f(x, y)] \tag{10.3-1}$$

where $f(x, y)$ is the gray level of point (x, y) and $p(x, y)$ denotes some local property of this point—for example, the average gray level of a neighborhood centered on (x, y). A thresholded image $g(x, y)$ is defined as

$$g(x, y) = \begin{cases} 1 & \text{if } f(x, y) > T \\ 0 & \text{if } f(x, y) \leq T. \end{cases} \tag{10.3-2}$$

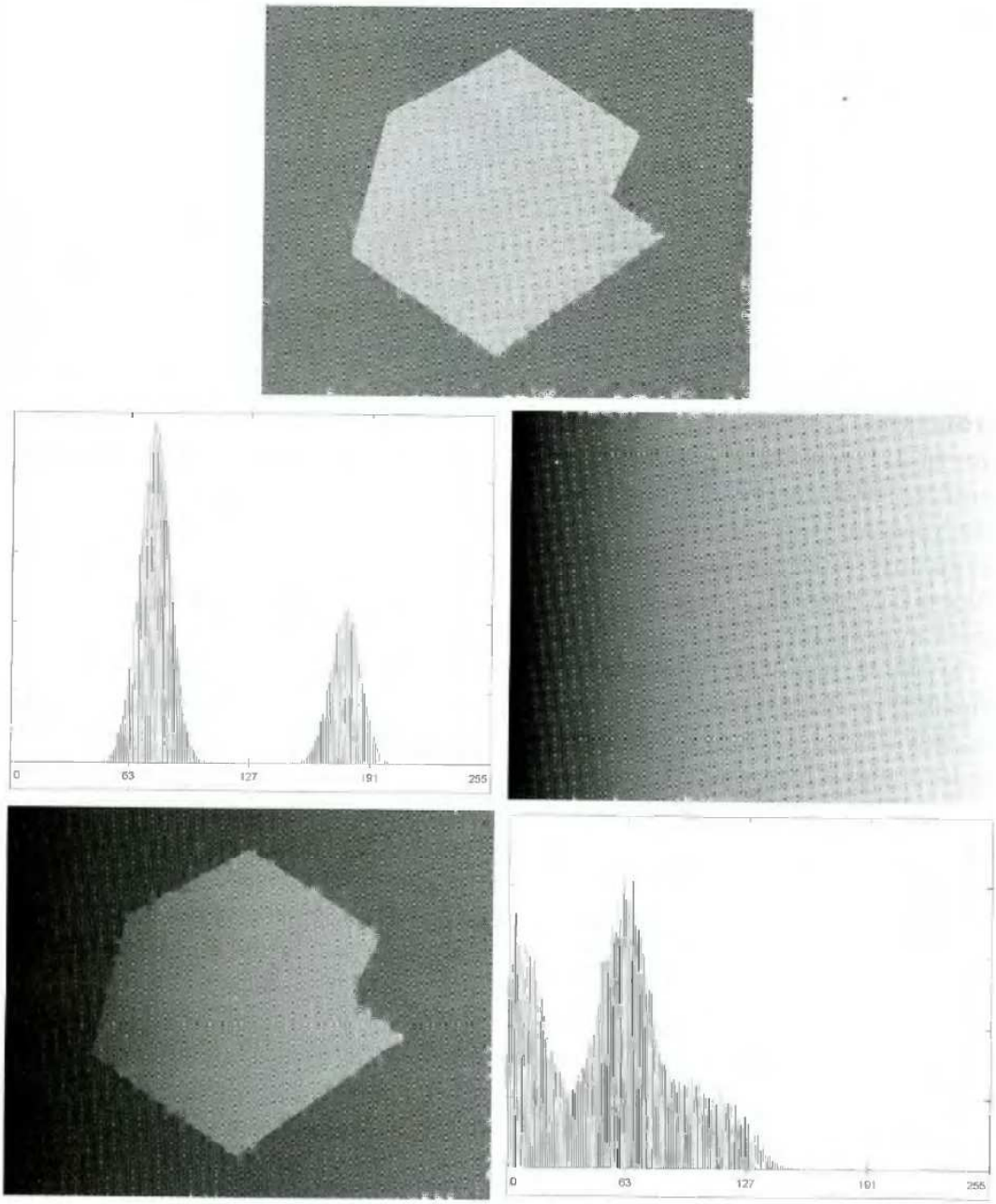
Thus, pixels labeled 1 (or any other convenient gray level) correspond to objects, whereas pixels labeled 0 (or any other gray level not assigned to objects) correspond to the background.

When T depends only on $f(x, y)$ (that is, only on gray-level values) the threshold is called *global*. If T depends on both $f(x, y)$ and $p(x, y)$, the threshold is called *local*. If, in addition, T depends on the spatial coordinates x and y, the threshold is called *dynamic* or *adaptive*.

10.3.2 The Role of Illumination

In Section 2.3.4 we introduced a simple model in which an image $f(x, y)$ is formed as the product of a reflectance component $r(x, y)$ and an illumination component $i(x, y)$. The purpose of this section is to use this model to discuss briefly the effect of illumination on thresholding, especially on global thresholding.

Consider the computer generated reflectance function shown in Fig. 10.27(a). The histogram of this function, shown in Fig. 10.27(b), is clearly bimodal and could be partitioned easily by placing a single global threshold, T, in the histogram



a
b c
d e

FIGURE 10.27 (a) Computer generated reflectance function. (b) Histogram of reflectance function. (c) Computer generated illumination function. (d) Product of (a) and (c). (e) Histogram of product image.

valley. Multiplying the reflectance function in Fig. 10.27(a) by the illumination function shown in Fig. 10.27(c) yields the image shown in Fig. 10.27(d). Figure 10.27(e) shows the histogram of this image. Note that the original valley was virtually eliminated, making segmentation by a single threshold an impossible task. Although we seldom have the reflectance function by itself to work with, this simple illustration shows that the reflective nature of objects and background could be such that they are easily separable. However, the image resulting from poor (in this case nonuniform) illumination could be quite difficult to segment.

The reason why the histogram in Fig. 10.27(e) is so distorted can be explained with aid of the discussion in Section 4.5. From Eq. (4.5-1),

$$f(x, y) = i(x, y)r(x, y). \tag{10.3-3}$$

Taking the natural logarithm of this equation yields a sum:

$$\begin{aligned} z(x, y) &= \ln f(x, y) \\ &= \ln i(x, y) + \ln r(x, y) \\ &= i'(x, y) + r'(x, y). \end{aligned} \tag{10.3-4}$$

From probability theory (Papoulis [1991]), if $i'(x, y)$ and $r'(x, y)$ are independent random variables, the histogram of $z(x, y)$ is given by the convolution of the histograms of $i'(x, y)$ and $r'(x, y)$. If $i(x, y)$ were constant, $i'(x, y)$ would be constant also, and its histogram would be a simple spike (like an impulse). The convolution of this impulselike function with the histogram of $r'(x, y)$ would leave the basic shape of this histogram unchanged (recall from the discussion in Section 4.2.4 that convolution of a function with an impulse copies the function at the location of the impulse). But if $i'(x, y)$ had a broader histogram (resulting from nonuniform illumination), the convolution process would smear the histogram of $r'(x, y)$, yielding a histogram for $z(x, y)$ whose shape could be quite different from that of the histogram of $r'(x, y)$. The degree of distortion depends on the broadness of the histogram of $i'(x, y)$, which in turn depends on the nonuniformity of the illumination function.

We have dealt with the logarithm of $f(x, y)$, instead of dealing with the image function directly, but the essence of the problem is clearly explained by using the logarithm to separate the illumination and reflectance components. This approach allows histogram formation to be viewed as a convolution process, thus explaining why a distinct valley in the histogram of the reflectance function could be smeared by improper illumination.

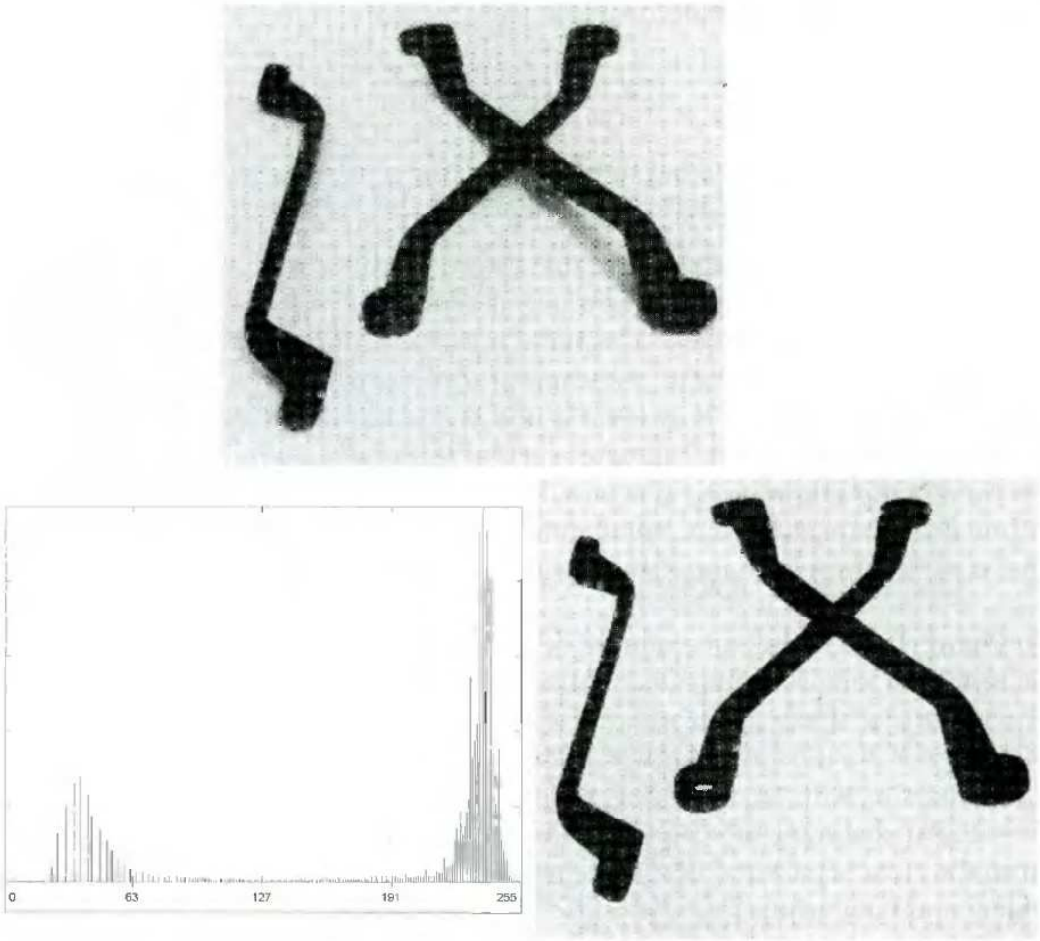
When access to the illumination source is available, a solution frequently used in practice to compensate for nonuniformity is to project the illumination pattern onto a constant, white reflective surface. This yields an image $g(x, y) = ki(x, y)$, where k is a constant that depends on the surface and $i(x, y)$ is the illumination pattern. Then, for any image $f(x, y) = i(x, y)r(x, y)$ obtained with the same illumination function, simply dividing $f(x, y)$ by $g(x, y)$ yields a normalized function $h(x, y) = f(x, y)/g(x, y) = r(x, y)/k$. Thus, if $r(x, y)$ can be segmented by using a single threshold T, then $h(x, y)$ can be segmented by using a single threshold of value T/k.

10.3.3 Basic Global Thresholding

With reference to the discussion in Section 10.3.1, the simplest of all thresholding techniques is to partition the image histogram by using a single global threshold, T, as illustrated in Fig. 10.26(a). Segmentation is then accomplished by scanning the image pixel by pixel and labeling each pixel as object or background, depending on whether the gray level of that pixel is greater or less than the value of T. As indicated earlier, the success of this method depends entirely on how well the histogram can be partitioned.

EXAMPLE 10.10: Global thresholding.

■ Figure 10.28(a) shows a simple image, and Fig. 10.28(b) shows its histogram. Figure 10.28(c) shows the result of segmenting Fig. 10.28(a) by using a threshold T midway between the maximum and minimum gray levels. This threshold



a
b c

FIGURE 10.28 (a) Original image. (b) Image histogram. (c) Result of global thresholding with T midway between the maximum and minimum gray levels.

achieved a "clean" segmentation by eliminating the shadows and leaving only the objects themselves. The objects of interest in this case are darker than the background, so any pixel with a gray level $\leq T$ was labeled black (0), and any pixel with a gray level $>T$ was labeled white (255). The key objective is merely to generate a binary image, so the black–white relationship could be reversed.

The type of global thresholding just described can be expected to be successful in highly controlled environments. One of the areas in which this often is possible is in industrial inspection applications, where control of the illumination usually is feasible. ■

The threshold in the preceding example was specified by using a heuristic approach, based on visual inspection of the histogram. The following algorithm can be used to obtain T automatically:

1. Select an initial estimate for T.
2. Segment the image using T. This will produce two groups of pixels: G_1 consisting of all pixels with gray level values $>T$ and G_2 consisting of pixels with values $<T$.
3. Compute the average gray level values μ_1 and μ_2 for the pixels in regions G_1 and G_2.

4. Compute a new threshold value:

$$T = \frac{1}{2}(\mu_1 + \mu_2).$$

5. Repeat steps 2 through 4 until the difference in T in successive iterations is smaller than a predefined parameter T_o.

When there is reason to believe that the background and object occupy comparable areas in the image, a good initial value for T is the average gray level of the image. When objects are small compared to the area occupied by the background (or vice versa), then one group of pixels will dominate the histogram and the average gray level is not as good an initial choice. A more appropriate initial value for T in cases such as this is a value midway between the maximum and minimum gray levels. The parameter T_o is used to stop the algorithm after changes become small in terms of this parameter. This is used when speed of iteration is an important issue.

EXAMPLE 10.11: Image segmentation using an estimated global threshold.

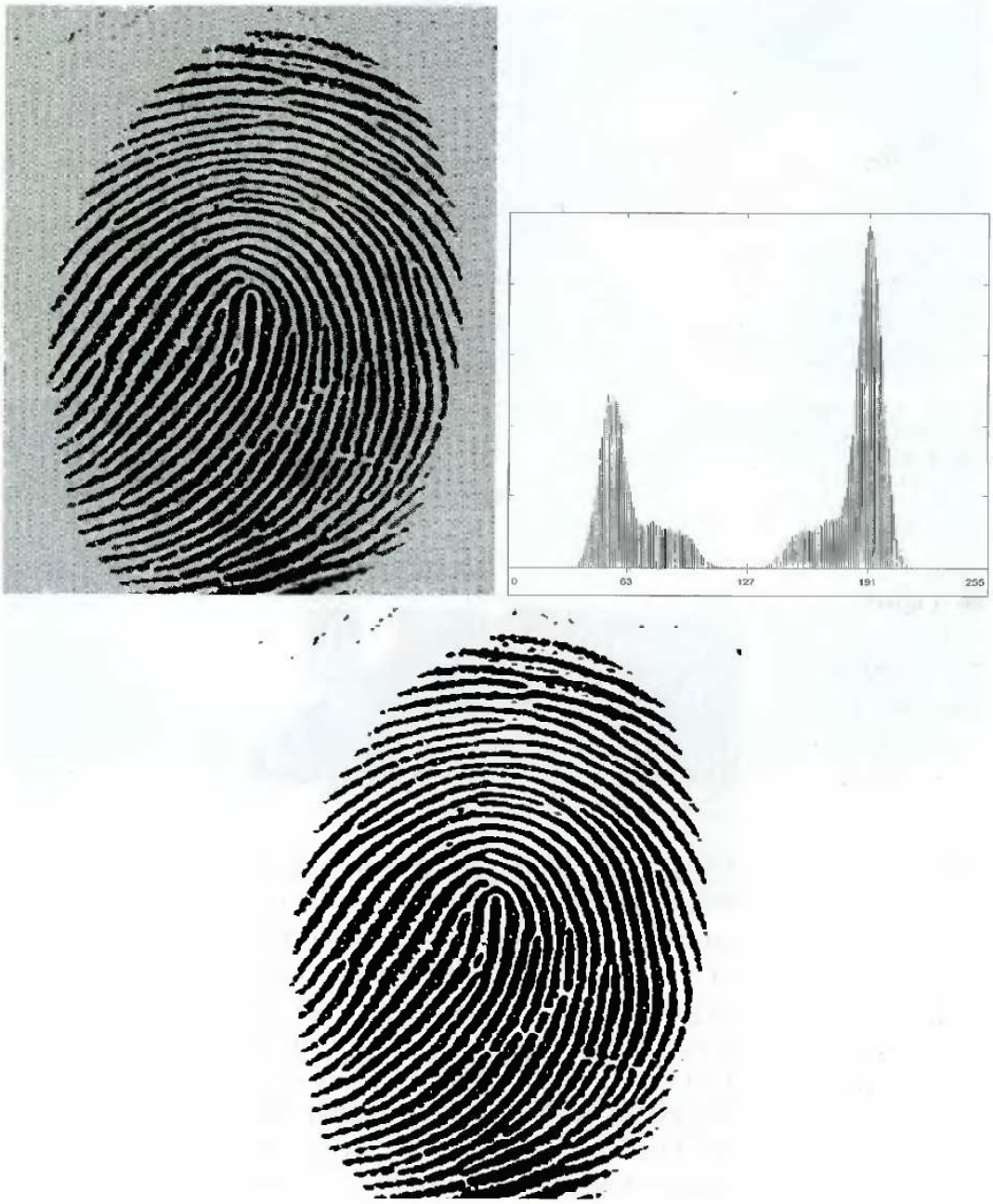
■ Figure 10.29 shows an example of segmentation based on a threshold estimated using the preceding algorithm. Figure 10.29(a) is the original image, and Fig. 10.29(b) is the image histogram. Note the clear valley of the histogram. Application of the iterative algorithm resulted in a value of 125.4 after three iterations starting with the average gray level and $T_o = 0$. The result obtained using $T = 125$ to segment the original image is shown in Fig. 10.29(c). As expected from the clear separation of modes in the histogram, the segmentation between object and background was very effective. ■

10.3.4 Basic Adaptive Thresholding

As illustrated in Fig. 10.27, imaging factors such as uneven illumination can transform a perfectly segmentable histogram into a histogram that cannot be partitioned effectively by a single global threshold. An approach for handling such a situation is to divide the original image into subimages and then utilize a different threshold to segment each subimage. The key issues in this approach are how to subdivide the image and how to estimate the threshold for each resulting subimage. Since the threshold used for each pixel depends on the location of the pixel in terms of the subimages, this type of thresholding is adaptive. We illustrate adaptive thresholding with a simple example. A more comprehensive example is given in the next section.

EXAMPLE 10.12: Basic adaptive thresholding.

■ Figure 10.30(a) shows the image from Fig. 10.27(d), which we concluded could not be thresholded effectively with a single global threshold. In fact, Fig. 10.30(b) shows the result of thresholding the image with a global threshold manually placed in the valley of its histogram [see Fig. 10.27(e)]. One approach to reduce the effect of nonuniform illumination is to subdivide the image into smaller subimages, such that the illumination of each subimage is approximately uniform. Figure 10.30(c) shows such a partition, obtained by subdividing the image into four equal parts, and then subdividing each part by four again.

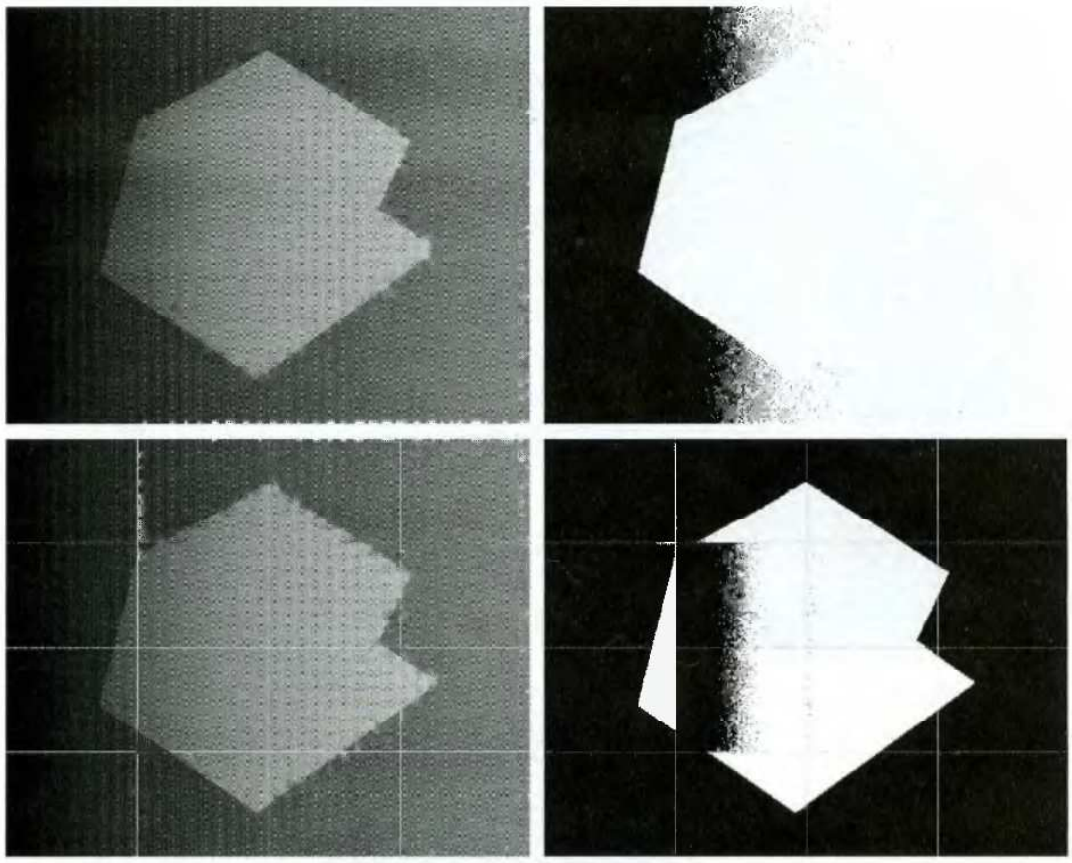


a b
c

FIGURE 10.29 (a) Original image. (b) Image histogram. (c) Result of segmentation with the threshold estimated by iteration. (Original courtesy of the National Institute of Standards and Technology.)

All the subimages that did not contain a boundary between object and background had variances of less than 75. All subimages containing boundaries had variances in excess of 100. Each subimage with variance greater than 100 was segmented with a threshold computed for that subimage using the algorithm discussed in the previous section. The initial value for T in each case was selected as the point midway between the minimum and maximum gray levels in the subimage. All subimages with variance less than 100 were treated as one composite image, which was segmented using a single threshold estimated using the same algorithm.

The result of segmentation using this procedure is shown in Fig. 10.30(d). With the exception of two subimages, the improvement over Fig. 10.30(b) is



a b
c d

FIGURE 10.30 (a) Original image. (b) Result of global thresholding. (c) Image subdivided into individual subimages. (d) Result of adaptive thresholding.

evident. The boundary between object and background in each of the improperly segmented subimages was small and dark, and the resulting histogram was almost unimodal. Figure 10.31(a) shows the top improperly segmented subimage from Fig. 10.30(c) and the subimage directly above it, which was segmented properly. The histogram of the subimage that was properly segmented is clearly bimodal, with well-defined peaks and valley. The other histogram is almost unimodal, with no clear distinction between object and background.

Figure 10.31(d) shows the failed subimage further subdivided into much smaller subimages, and Fig. 10.31(e) shows the histogram of the top, left small subimage. This subimage contains the transition between object and background. This smaller subimage has a clearly bimodal histogram and should be easily segmentable. This, in fact, is the case, as shown in Fig. 10.31(f). This figure also shows the segmentation of all the other small subimages. All these subimages had a nearly unimodal histogram, and their average gray level was closer to the object than to the background, so they were all classified as object. It is left as a project for the reader to show that considerably more accurate segmentation can be achieved by subdividing the entire image in Fig. 10.30(a) into subimages of the size shown in Fig. 10.31(d). ■

10.3.5 Optimal Global and Adaptive Thresholding

In this section we discuss a method for estimating thresholds that produce the minimum average segmentation error. As an illustration, the method is applied

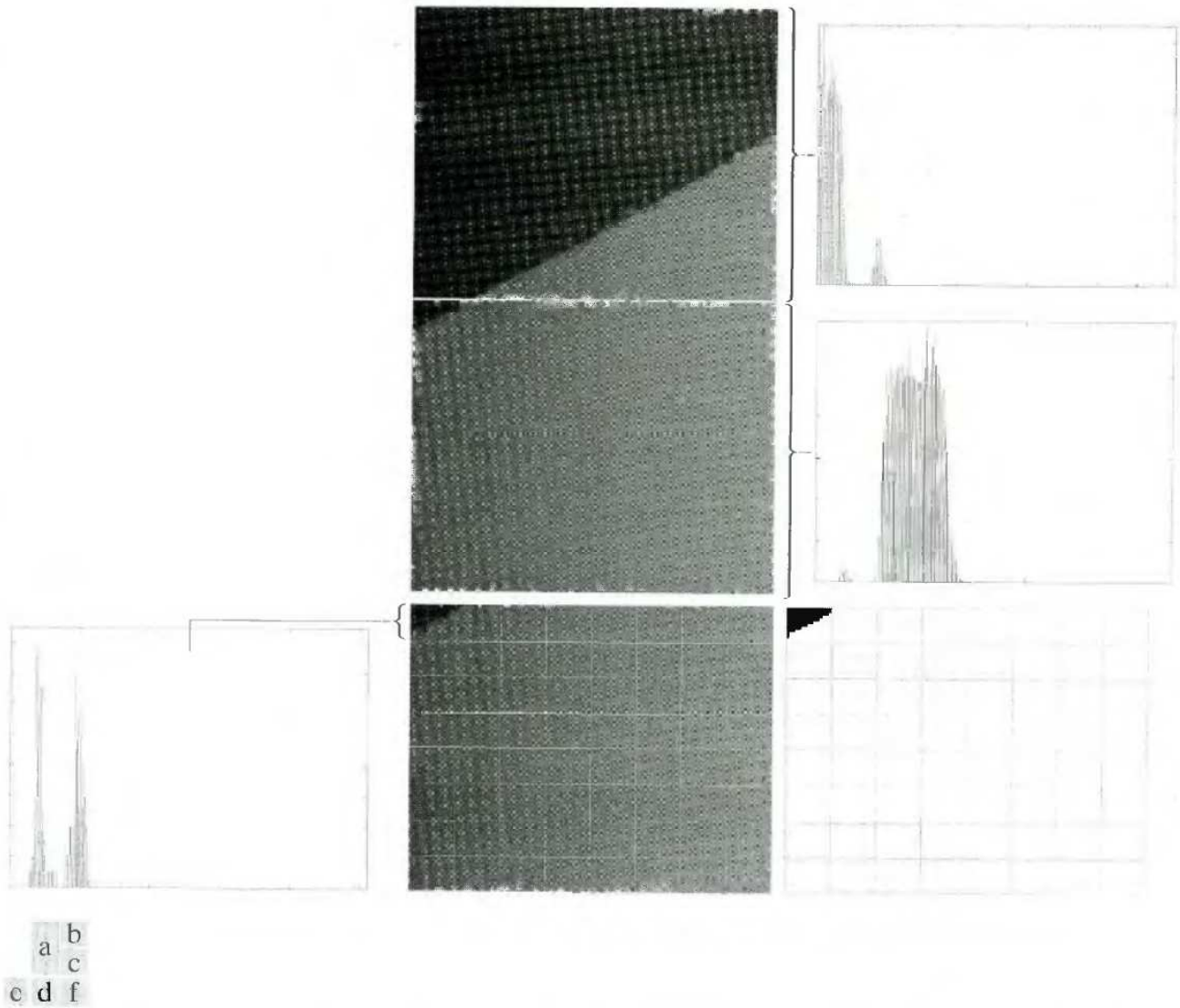


FIGURE 10.31 (a) Properly and improperly segmented subimages from Fig. 10.30. (b)–(c) Corresponding histograms. (d) Further subdivision of the improperly segmented subimage. (e) Histogram of small subimage at top, left. (f) Result of adaptively segmenting (d).

to a problem that requires solution of several important issues found frequently in the practical application of thresholding.

Suppose that an image contains only two principal gray-level regions. Let z denote gray-level values. We can view these values as random quantities, and their histogram may be considered an estimate of their probability density function (PDF), $p(z)$. This overall density function is the sum or mixture of two densities, one for the light and the other for the dark regions in the image. Furthermore, the mixture parameters are proportional to the relative areas of the dark and light regions. If the form of the densities is known or assumed, it is possible to determine an optimal threshold (in terms of minimum error) for segmenting the image into the two distinct regions.

See inside front cover

Consult the book web site for a brief review of probability theory.

Figure 10.32 shows two probability density functions. Assume that the larger of the two PDFs corresponds to the background levels while the smaller one

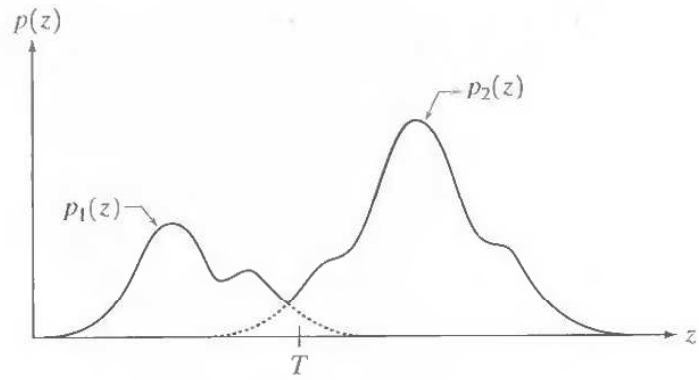


FIGURE 10.32 Gray-level probability density functions of two regions in an image.

describes the gray levels of objects in the image. The mixture probability density function describing the overall gray-level variation in the image is

$$p(z) = P_1 p_1(z) + P_2 p_2(z). \tag{10.3-5}$$

Here, P_1 and P_2 are the probabilities of occurrence of the two classes of pixels; that is, P_1 is the probability (a number) that a random pixel with value z is an object pixel. Similarly, P_2 is the probability that the pixel is a background pixel. We are assuming that any given pixel belongs either to an object or to the background, so that

$$P_1 + P_2 = 1. \tag{10.3-6}$$

An image is segmented by classifying as background all pixels with gray levels greater than a threshold T (see Fig. 10.32). All other pixels are called object pixels. Our main objective is to select the value of T that minimizes the average error in making the decisions that a given pixel belongs to an object or to the background.

Recall that the probability of a random variable having a value in the interval $[a, b]$ is the integral of its probability density function from a to b, which is the area of the PDF curve between these two limits. Thus, the probability of *erroneously* classifying a background point as an object point is

$$E_1(T) = \int_{-\infty}^{T} p_2(z)\, dz. \tag{10.3-7}$$

This is the area under the curve of $p_2(z)$ to the left of the threshold. Similarly, the probability of erroneously classifying an object point as background is

$$E_2(T) = \int_{T}^{\infty} p_1(z)\, dz, \tag{10.3-8}$$

which is the area under the curve of $p_1(z)$ to the right of T. Then the overall probability of error is

$$E(T) = P_2 E_1(T) + P_1 E_2(T). \tag{10.3-9}$$

Note how the quantities E_1 and E_2 are weighted (given importance) by the probability of occurrence of object or background pixels. Note also that the sub-

scripts are opposites. This is simple to explain. Consider, for example, the extreme case in which background points are known never to occur. In this case $P_2 = 0$. The contribution to the overall error (E) of classifying a background point as an object point (E_1) should be zeroed out because background points are known never to occur. This is accomplished by multiplying E_1 by $P_2 = 0$. If background and object points are equally likely to occur, then the weights are $P_1 = P_2 = 0.5$.

To find the threshold value for which this error is minimal requires differentiating $E(T)$ with respect to T (using Leibniz's rule) and equating the result to 0. The result is

$$P_1 p_1(T) = P_2 p_2(T). \tag{10.3-10}$$

This equation is solved for T to find the optimum threshold. Note that if $P_1 = P_2$, then the optimum threshold is where the curves for $p_1(z)$ and $p_2(z)$ intersect (see Fig. 10.32).

Obtaining an analytical expression for T requires that we know the equations for the two PDFs. Estimating these densities in practice is not always feasible, and an approach used often is to employ densities whose parameters are reasonably simple to obtain. One of the principal densities used in this manner is the Gaussian density, which is completely characterized by two parameters: the mean and the variance. In this case,

$$p(z) = \frac{P_1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(z-\mu_1)^2}{2\sigma_1^2}} + \frac{P_2}{\sqrt{2\pi}\sigma_2} e^{-\frac{(z-\mu_2)^2}{2\sigma_2^2}} \tag{10.3-11}$$

where μ_1 and σ_1^2 are the mean and variance of the Gaussian density of one class of pixels (say, objects) and μ_2 and σ_2^2 are the mean and variance of the other class. Using this equation in the general solution of Eq. (10.3-10) results in the following solution for the threshold T:

$$AT^2 + BT + C = 0 \tag{10.3-12}$$

where

$$\begin{aligned} A &= \sigma_1^2 - \sigma_2^2 \\ B &= 2(\mu_1\sigma_2^2 - \mu_2\sigma_1^2) \\ C &= \sigma_1^2\mu_2^2 - \sigma_2^2\mu_1^2 + 2\sigma_1^2 2\sigma_2^2 \ln(\sigma_2 P_1/\sigma_1 P_2). \end{aligned} \tag{10.3-13}$$

Since a quadratic equation has two possible solutions, two threshold values may be required to obtain the optimal solution.

If the variances are equal, $\sigma^2 = \sigma_1^2 = \sigma_2^2$, a single threshold is sufficient:

$$T = \frac{\mu_1 + \mu_2}{2} + \frac{\sigma^2}{\mu_1 - \mu_2} \ln\left(\frac{P_2}{P_1}\right). \tag{10.3-14}$$

If $P_1 = P_2$, the optimal threshold is the average of the means. The same is true if $\sigma = 0$. Determining the optimal threshold may be similarly accomplished for other densities of known form, such as the Raleigh and log-normal densities.

Instead of assuming a functional form for $p(z)$, a minimum mean-square-error approach may be used to estimate a composite gray-level PDF of an image

from the image histogram. For example, the mean square error between the (continuos) mixture density $p(z)$ and the (discrete) image histogram $h(z_i)$ is

$$e_{\text{ms}} = \frac{1}{n} \sum_{i=1}^{n} [p(z_i) - h(z_i)]^2 \tag{10.3-15}$$

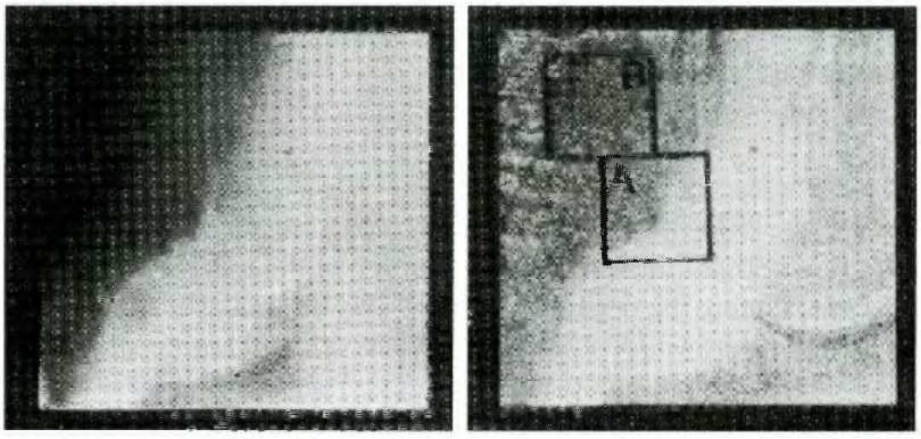
where an n-point histogram is assumed. The principal reason for estimating the complete density is to determine the presence or absence dominant modes in the PDF. For example, two dominant modes typically indicate the presence of edges in the image (or region) over which the PDF is computed.

In general, determining analytically the parameters that minimize this mean square error is not a simple matter. Even for the Gaussian case, the straightforward computation of equating the partial derivatives to 0 leads to a set of simultaneous transcendental equations that usually can be solved only by numerical procedures, such as a conjugate gradients or Newton's method for simultaneous nonlinear equations.

EXAMPLE 10.13: Use of optimum thresholding for image segmentation.

■ The following is one of the earliest (and still one of the most instructive) examples of segmentation by optimum thresholding in image processing. This example is particularly interesting at this junction because it shows how segmentation results can be improved by employing preprocessing techniques based on methods developed in our discussion of image enhancement. In addition, the example also illustrates the use of local histogram estimation and adaptive thresholding. The general problem is to outline automatically the boundaries of heart ventricles in cardioangiograms (X-ray images of a heart that has been injected with a contrast medium). The approach discussed here was developed by Chow and Kaneko [1972] for outlining boundaries of the left ventricle of the heart.

Prior to segmentation, all images were preprocessed as follows: (1) Each pixel was mapped with a log function (see Section 3.2.2) to counter exponential effects caused by radioactive absorption. (2) An image obtained before application of the contrast medium was subtracted from each image captured after the medium was injected in order to remove the spinal column present in both images (see Section 3.4.1). (3) Several angiograms were summed in order to reduce random noise (see Section 3.4.2). Figure 10.33 shows a cardioangiogram



a b

FIGURE 10.33 A cardioangiogram before and after preprocessing. (Chow and Kaneko.)

before and after preprocessing (an explanation of the regions marked A and B is given in the following paragraph).

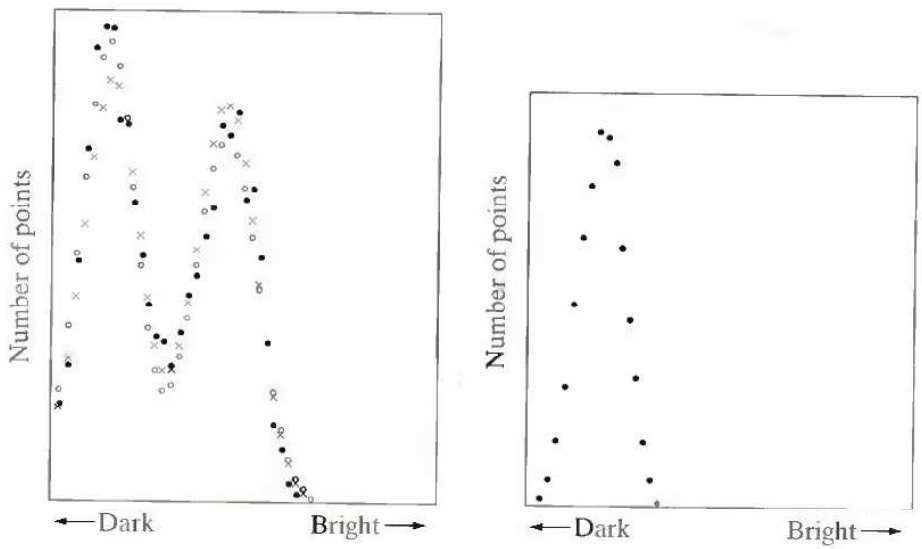
In order to compute the optimal thresholds, each preprocessed image was subdivided into 49 regions by placing a 7×7 grid with 50% overlap over each image (all original images shown in this example are of size 256×256 pixels). Each of the 49 resulting overlapped regions contained 64×64 pixels. Figures 10.34(a) and 10.34(b) are the histograms of the regions marked A and B in Fig. 10.33(b). Note that the histogram for region A clearly is bimodal, indicating the presence of a boundary. The histogram for region B, however, is unimodal, indicating the absence of two markedly distinct regions.

After all 49 histograms were computed, a test of bimodality was performed to reject the unimodal histograms. The remaining histograms were then fitted by bimodal Gaussian density curves [see Eq. (10.3-11)] using a conjugate gradient hill-climbing method to minimize the error function given in Eq. (10.3-15). The ×'s and ○'s in Fig. 10.34(a) are two fits to the histogram shown in black dots. The optimum thresholds were then obtained by using Eqs. (10.3-12) and (10.3-13).

At this stage of the process only the regions with bimodal histograms were assigned thresholds. The thresholds for the remaining regions were obtained by interpolating these thresholds. Then a second interpolation was carried out point by point by using neighboring threshold values so that, at the end of the procedure, every point in the image had been assigned a threshold. Finally, a binary decision was carried out for each pixel using the rule

$$f(x, y) = \begin{cases} 1 & \text{if } f(x, y) \geq T_{xy} \\ 0 & \text{otherwise} \end{cases}$$

where T_{xy} was the threshold assigned to location (x, y) in the image [note that these are adaptive thresholds, because they depend on the spatial coordinates (x, y)]. Boundaries were obtained by taking the gradient of the binary picture. Figure 10.35 shows the boundaries superimposed on the original image. Considering the variability and complexity of the images involved, this procedure yielded excellent segmentation results. ■



a b

FIGURE 10.34 Histograms (black dots) of (a) region A, and (b) region B in Fig. 10.33(b). (Chow and Kaneko.)

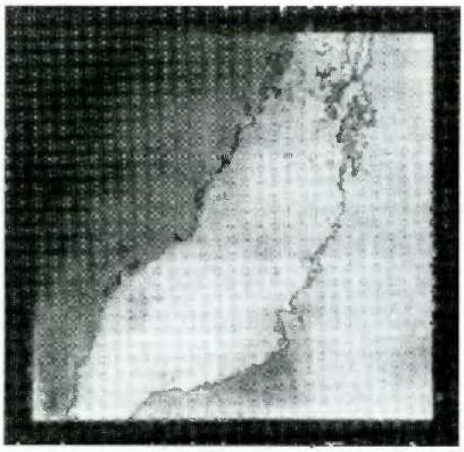


FIGURE 10.35 Cardioangiogram showing superimposed boundaries. (Chow and Kaneko.)

10.3.6 Use of Boundary Characteristics for Histogram Improvement and Local Thresholding

Based on the discussions in the previous five sections, it is intuitively evident that the chances of selecting a "good" threshold are enhanced considerably if the histogram peaks are tall, narrow, symmetric, and separated by deep valleys. One approach for improving the shape of histograms is to consider only those pixels that lie on or near the edges between objects and the background. An immediate and obvious improvement is that histograms would be less dependent on the relative sizes of objects and the background. For instance, the histogram of an image composed of a small object on a large background area (or vice versa) would be dominated by a large peak because of the high concentration of one type of pixels. Figures 10.30 and 10.31 are a good illustration of how segmentation performance is affected by this condition.

If only the pixels on or near the edge between object and the background were used, the resulting histogram would have peaks of approximately the same height. In addition, the probability that any of those given pixels lies on an object would be approximately equal to the probability that it lies on the background, thus improving the symmetry of the histogram peaks. Finally, as indicated in the following paragraph, using pixels that satisfy some simple measures based on gradient and Laplacian operators has a tendency to deepen the valley between histogram peaks.

The principal problem with the approach just discussed is the implicit assumption that the edges between objects and background are known. This information clearly is not available during segmentation, as finding a division between objects and background is precisely what segmentation is all about. However, from the discussion in Section 10.1.3, an indication of whether a pixel is on an edge may be obtained by computing its gradient. In addition, use of the Laplacian can yield information regarding whether a given pixel lies on the dark or light side of an edge. The average value of the Laplacian is 0 at the transition of an edge (see Fig. 10.6), so in practice the valleys of histograms formed from the pixels selected by a gradient/Laplacian criterion can be expected to be sparsely populated. This property produces the highly desirable deep valleys discussed previously.

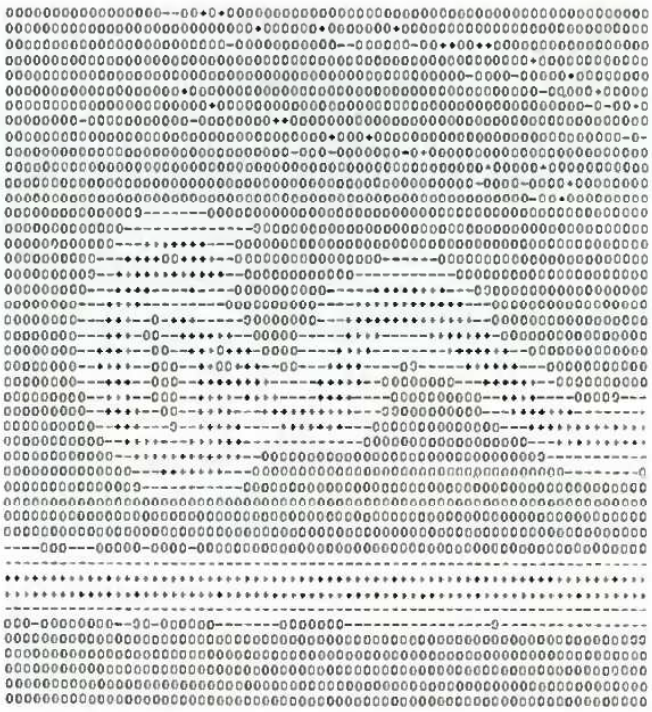


FIGURE 10.36 Image of a handwritten stroke coded by using Eq. (10.3-16). (Courtesy of IBM Corporation.)

The gradient ∇f at any point (x, y) in an image is given by Eq. (10.1-4) or (10.1-12). Similarly, the Laplacian $\nabla^2 f$ is given by Eq. (10.1-14) or (10.1-15). These two quantities may be used to form a three-level image, as follows:

$$s(x, y) = \begin{cases} 0 & \text{if } \nabla f < T \\ + & \text{if } \nabla f \geq T \quad \text{and} \quad \nabla^2 f \geq 0 \\ - & \text{if } \nabla f \geq T \quad \text{and} \quad \nabla^2 f < 0 \end{cases} \tag{10.3-16}$$

where the symbols 0, +, and − represent any three distinct gray levels, T is a threshold, and the gradient and Laplacian are computed at every point (x, y). For a dark object on a light background, and with reference to Fig. 10.6, the use of Eq. (10.3-16) produces an image $s(x, y)$ in which (1) all pixels that are not on an edge (as determined by ∇f being less than T) are labeled 0; (2) all pixels on the dark side of an edge are labeled +; and (3) all pixels on the light side of an edge are labeled −. The symbols + and − in Eq. (10.3-16) are reversed for a light object on a dark background. Figure 10.36 shows the labeling produced by Eq. (10.3-16) for an image of a dark, underlined stroke written on a light background.

The information obtained with this procedure can be used to generate a segmented, binary image in which 1's correspond to objects of interest and 0's correspond to the background. The transition (along a horizontal or vertical scan line) from a light background to a dark object must be characterized by the occurrence of a − followed by a + in $s(x, y)$. The interior of the object is composed of pixels that are labeled either 0 or +. Finally, the transition from the object back to the background is characterized by the occurrence of a + followed by a −. Thus a horizontal or vertical scan line containing a section of an object has the following structure:

$$(\cdots)(-, +)(0 \text{ or } +)(+, -)(\cdots)$$

a
b

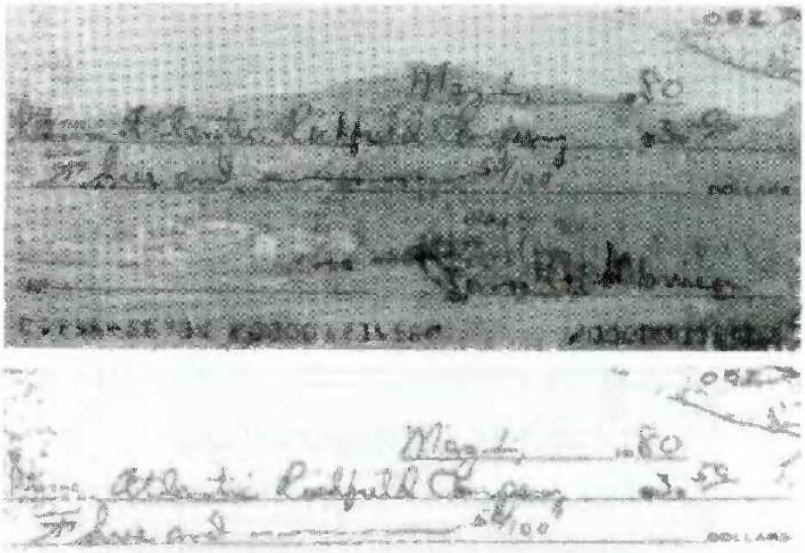


FIGURE 10.37 (a) Original image. (b) Image segmented by local thresholding. (Courtesy of IBM Corporation.)

where $(\cdots)$ represents any combination of $+$, $-$, and 0. The innermost parentheses contain object points and are labeled 1. All other pixels along the same scan line are labeled 0, with the exception of any other sequence of (0 or $+$) bounded by $(-, +)$ and $(+, -)$.

EXAMPLE 10.14: Image segmentation by local thresholding.

■ Figure 10.37(a) shows an image of an ordinary scenic bank check. Figure 10.38 shows the histogram as a function of gradient values for pixels with gradients greater than 5. Note that this histogram has two dominant modes that are symmetric, nearly of the same height, and are separated by a distinct valley. Finally, Fig. 10.37(b) shows the segmented image obtained by using Eq. (10.3-16) with T at or near the midpoint of the valley. The result was made binary by using the sequence analysis just discussed. Note that this example is an illustration of local thresholding, as defined in Eq. (10.3-1), because the value of T was determined from a histogram of the gradient and Laplacian, which are local properties. ■

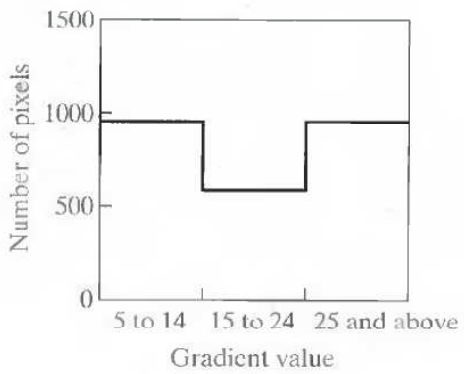


FIGURE 10.38 Histogram of pixels with gradients greater than 5. (Courtesy of IBM Corporation.)

10.3.7 Thresholds Based on Several Variables

So far we have been concerned with thresholding gray levels. In some cases, a sensor can make available more than one variable to characterize each pixel in an image, and thus allow *multispectral theresholding*. As discussed in some detail in Section 6.7, color imaging is a good example, in which each pixel is characterized by three RGB values. In this case, constructing a 3-D "histogram" becomes possible. The basic procedure is analogous to the method used for one variable. For example, for an image with three variables (RGB components), each having 16 possible levels, a $16 \times 16 \times 16$ grid (cube) is formed. Inserted in each cell of the cube is the number of pixels whose RGB components have values corresponding to the coordinates defining the location of that particular cell. Each entry is then divided by the total number of pixels in the image to form a normalized histogram.

The concept of thresholding now becomes one of finding clusters of points in 3-D space. Suppose, for example, that K significant clusters of points are found in the histogram. The image can be segmented by assigning one arbitrary value (say, white) to pixels whose RGB components are closer to one cluster and another value (say, black) to the other pixels in the image. This concept is easily extendable to more components and certainly to more clusters. The principal difficulty is that cluster seeking becomes an increasingly complex task as the number of variables increases. Cluster-seeking methods can be found, for example, in the books by Duda, Hart, and Stork [2001], and Tou and Gonzalez [1974].

EXAMPLE 10.15: Multispectral thresholding.

■ The image shown in Fig. 10.39(a) is a monochrome picture of a color photograph. The original color image is composed of three 16-level RGB images. The scarf is a vivid red, and the hair and facial colors are light and different in spectral characteristics from the window and other background features.

Figure 10.39(b) was obtained by thresholding about one of the histogram clusters corresponding to facial tones. Note that the window, which in the monochrome

a b c

FIGURE 10.39 (a) Original color image shown as a monochrome picture. (b) Segmentation of pixels with colors close to facial tones. (c) Segmentation of red components.

picture is close in gray-level value to the hair, does not appear in the segmented image because of the use of multispectral characteristics to separate these two regions. Figure 10.39(c) was obtained by thresholding about a cluster close to the red axis. In this case only the scarf and part of a flower (which is red also) appeared in the segmented result. The threshold used to obtain both results was a distance of one cell. Thus any pixel whose components were outside the cell enclosing the center of the cluster in question was classified as background (black). Pixels whose components placed them inside the cell were coded white. ■

As discussed in Section 6.7, color segmentation can be based on any of the color models introduced in Chapter 6. For instance, hue and saturation are important properties in numerous applications dealing with the use of imaging for automated inspection. These properties are particularly important in attempts to emulate the equivalent function performed by humans, such as in the inspection of fruits for ripeness or in the inspection of manufactured goods. As mentioned in Chapter 6, the Hue, Saturation, Intensity (HSI) model is ideal for these types of applications because it is closely related to the way in which humans describe the perception of color. A segmentation approach using the hue and saturation components of a color signal also is particularly attractive, because it involves 2-D data clusters that are easier to analyze than, say, the 3-D clusters needed for RGB segmentation.

10.4 Region-Based Segmentation

The objective of segmentation is to partition an image into regions. In Sections 10.1 and 10.2 we approached this problem by finding boundaries between regions based on discontinuities in gray levels, whereas in Section 10.3 segmentation was accomplished via thresholds based on the distribution of pixel properties, such as gray-level values or color. In this section we discuss segmentation techniques that are based on finding the regions directly.

10.4.1 Basic Formulation

Let R represent the entire image region. We may view segmentation as a process that partitions R into n subregions, $R_1, R_2, \ldots, R_n$, such that

(a) $\bigcup_{i=1}^{n} R_i = R$.
(b) R_i is a connected region, $i = 1, 2, \ldots, n$.
(c) $R_i \cap R_j = \varnothing$ for all i and $j, i \neq j$.
(d) $P(R_i) = \text{TRUE}$ for $i = 1, 2, \ldots, n$.
(e) $P(R_i \cup R_j) = \text{FALSE}$ for $i \neq j$.

Here, $P(R_i)$ is a logical predicate defined over the points in set R_i and $\varnothing$ is the null set.

Condition (a) indicates that the segmentation must be complete; that is, every pixel must be in a region. Condition (b) requires that points in a region must be connected in some predefined sense (see Section 2.5.2 regarding connectivity).

Condition (c) indicates that the regions must be disjoint. Condition (d) deals with the properties that must be satisfied by the pixels in a segmented region—for example $P(R_i)$ = TRUE if all pixels in R_i have the same gray level. Finally, condition (e) indicates that regions R_i and R_j are different in the sense of predicate P.

10.4.2 Region Growing

As its name implies, *region growing* is a procedure that groups pixels or subregions into larger regions based on predefined criteria. The basic approach is to start with a set of "seed" points and from these grow regions by appending to each seed those neighboring pixels that have properties similar to the seed (such as specific ranges of gray level or color).

Selecting a set of one or more starting points often can be based on the nature of the problem, as will be shown in Example 10.16. When a priori information is not available, the procedure is to compute at every pixel the same set of properties that ultimately will be used to assign pixels to regions during the growing process. If the result of these computations shows clusters of values, the pixels whose properties place them near the centroid of these clusters can be used as seeds.

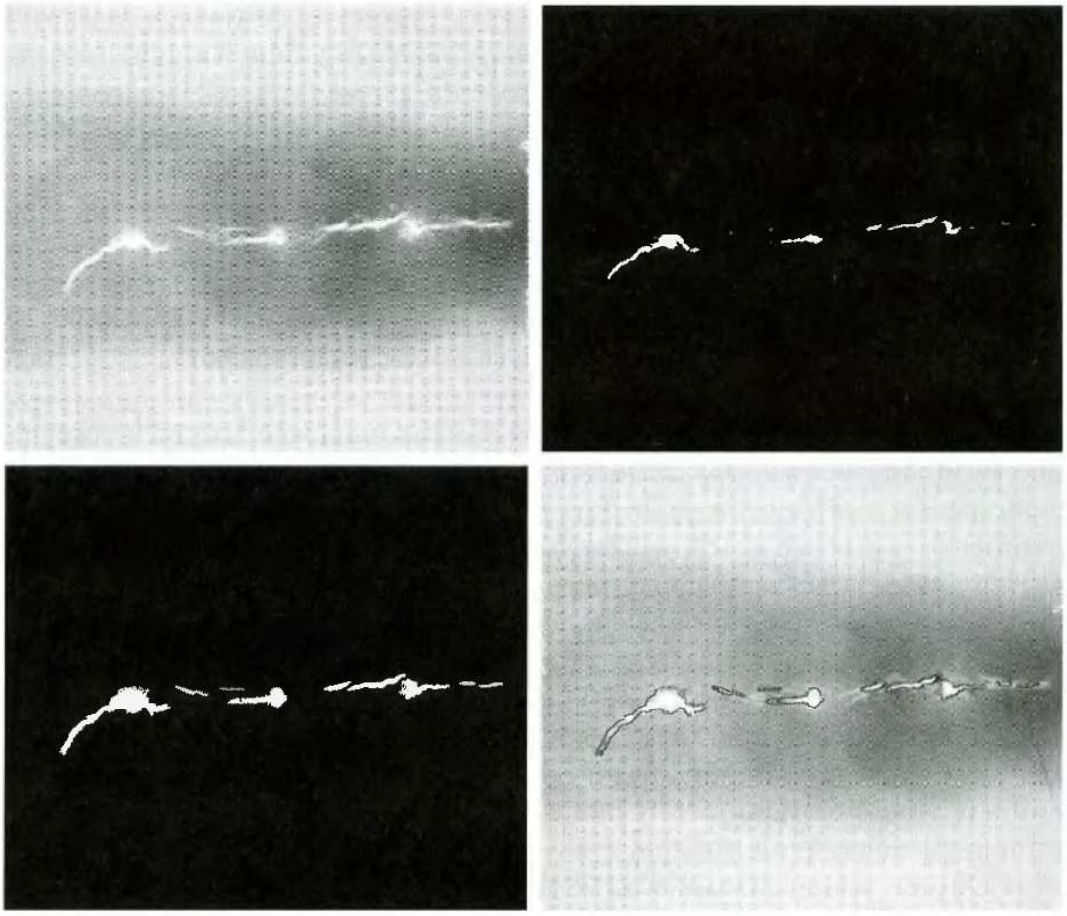
The selection of similarity criteria depends not only on the problem under consideration, but also on the type of image data available. For example, the analysis of land-use satellite imagery depends heavily on the use of color. This problem would be significantly more difficult, or even impossible, to handle without the inherent information available in color images. When the images are monochrome, region analysis must be carried out with a set of descriptors based on gray levels and spatial properties (such as moments or texture). We discuss descriptors useful for region characterization in Chapter 11.

Descriptors alone can yield misleading results if connectivity or adjacency information is not used in the region-growing process. For example, visualize a random arrangement of pixels with only three distinct gray-level values. Grouping pixels with the same gray level to form a "region" without paying attention to connectivity would yield a segmentation result that is meaningless in the context of this discussion.

Another problem in region growing is the formulation of a stopping rule. Basically, growing a region should stop when no more pixels satisfy the criteria for inclusion in that region. Criteria such as gray level, texture, and color, are local in nature and do not take into account the "history" of region growth. Additional criteria that increase the power of a region-growing algorithm utilize the concept of size, likeness between a candidate pixel and the pixels grown so far (such as a comparison of the gray level of a candidate and the average gray level of the grown region), and the shape of the region being grown. The use of these types of descriptors is based on the assumption that a model of expected results is at least partially available.

EXAMPLE 10.16: Application of region growing in weld inspection.

■ Figure 10.40(a) shows an X-ray image of a weld (the horizontal dark region) containing several cracks and porosities (the bright, white streaks running horizontally through the middle of the image). We wish to use region growing to segment the regions of the weld failures. These segmented features could be used



a b
c d

FIGURE 10.40 (a) Image showing defective welds. (b) Seed points. (c) Result of region growing. (d) Boundaries of segmented defective welds (in black). (Original image courtesy of X-TEK Systems, Ltd.).

for inspection, for inclusion in a database of historical studies, for controlling an automated welding system, and for other numerous applications.

The first order of business is to determine the initial seed points. In this application, it is known that pixels of defective welds tend to have the maximum allowable digital value (255 in this case). Based on this information, we selected as starting points all pixels having values of 255. The points thus extracted from the original image are shown in Fig. 10.40(b). Note that many of the points are clustered into *seed regions*.

The next step is to choose criteria for region growing. In this particular example we chose two criteria for a pixel to be annexed to a region: (1) The absolute gray-level difference between any pixel and the seed had to be less than 65. This number is based on the histogram shown in Fig. 10.41 and represents the difference between 255 and the location of the first major valley to the left, which is representative of the highest gray level value in the dark weld region. (2) To be included in one of the regions, the pixel had to be 8-connected to at least one pixel in that region. If a pixel was found to be connected to more than one region, the regions were merged.

Figure 10.40(c) shows the regions that resulted by starting with the seeds in Fig. 10.40(b) and utilizing the criteria defined in the previous paragraph. Superimposing the boundaries of these regions on the original image [Fig. 10.40(d)] reveals that the region-growing procedure did indeed segment the defective

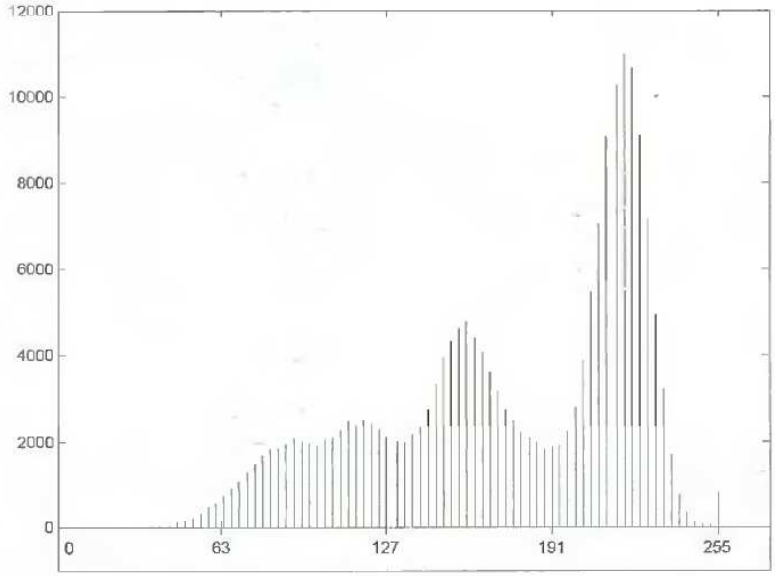


FIGURE 10.41 Histogram of Fig. 10.40(a).

welds with an acceptable degree of accuracy. It is of interest to note that it was not necessary to specify any stopping rules in this case because the criteria for region growing were sufficient to isolate the features of interest. ■

It was mentioned in Section 10.3.1 in connection with Fig. 10.26(b) that problems having multimodal histograms generally are best solved using region-based approaches. The histogram shown in Fig. 10.41 is an excellent example of a "clean" multimodal histogram. This histogram and the results in Example 10.16 confirm the assertion that, even with well-behaved histograms, multilevel thresholding is a difficult proposition. Based on the results of Example 10.16, it should be intuitively obvious that this problem cannot be solved effectively by any reasonably general method of automatic threshold selection based on gray levels alone. The use of *connectivity* was fundamental in solving the problem.

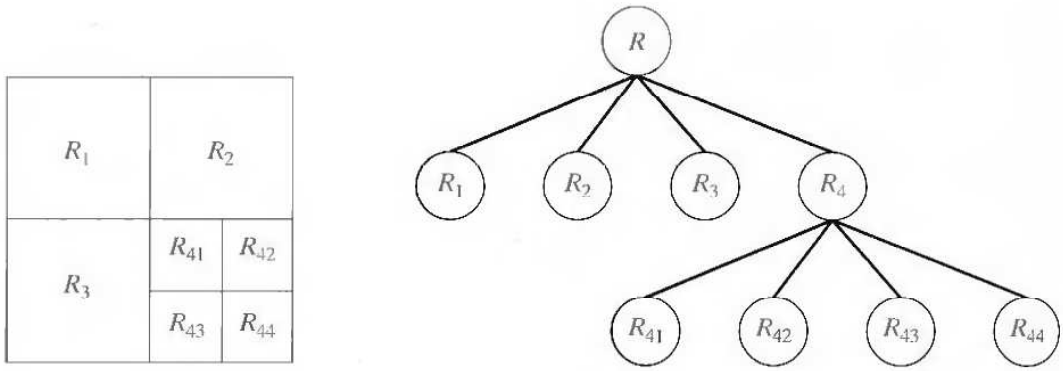
10.4.3 Region Splitting and Merging

The procedure just discussed grows regions from a set of seed points. An alternative is to subdivide an image initially into a set of arbitrary, disjointed regions and then merge and/or split the regions in an attempt to satisfy the conditions stated in Section 10.4.1. A split and merge algorithm that iteratively works toward satisfying these constraints is developed next.

Let R represent the entire image region and select a predicate P. One approach for segmenting R is to subdivide it successively into smaller and smaller quadrant regions so that, for any region R_i, $P(R_i) = \text{TRUE}$. We start with the entire region. If $P(R) = \text{FALSE}$, we divide the image into quadrants. If P is FALSE for any quadrant, we subdivide that quadrant into subquadrants, and so on. This particular splitting technique has a convenient representation in the form of a so-called *quadtree* (that is, a tree in which nodes have exactly four descendants), as illustrated in Fig. 10.42. Note that the root of the tree corresponds to the entire image and that each node corresponds to a subdivision. In this case, only R_4 was subdivided further.

a b

FIGURE 10.42
(a) Partitioned image.
(b) Corresponding quadtree.



If only splitting were used, the final partition likely would contain adjacent regions with identical properties. This drawback may be remedied by allowing merging, as well as splitting. Satisfying the constraints of Section 10.4.1 requires merging only adjacent regions whose combined pixels satisfy the predicate P. That is, two adjacent regions R_j and R_k are merged only if $P(R_j \cup R_k) = \text{TRUE}$.

The preceding discussion may be summarized by the following procedure, in which, at any step we

1. Split into four disjoint quadrants any region R_i for which $P(R_i) = \text{FALSE}$.
2. Merge any adjacent regions R_j and R_k for which $P(R_j \cup R_k) = \text{TRUE}$.
3. Stop when no further merging or splitting is possible.

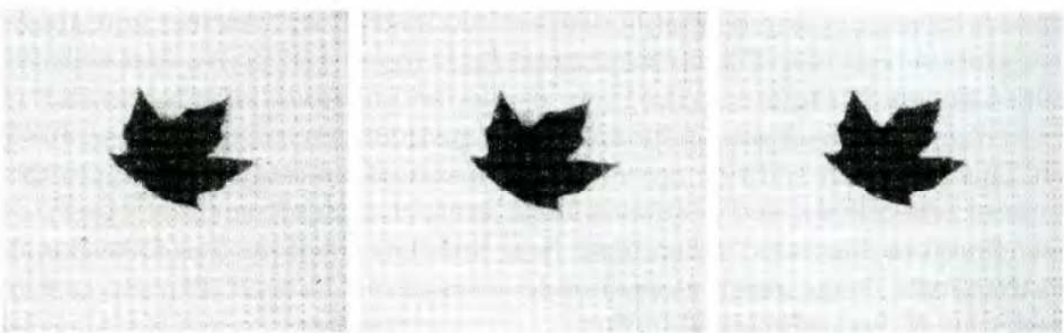
Several variations of the preceding basic theme are possible. For example, one possibility is to split the image initially into a set of blocks. Further splitting is carried out as described previously, but merging is initially limited to groups of four blocks that are descendants in the quadtree representation and that satisfy the predicate P. When no further mergings of this type are possible, the procedure is terminated by one final merging of regions satisfying step 2. At this point, the merged regions may be of different sizes. The principal advantage of this approach is that it uses the same quadtree for splitting and merging, until the final merging step.

EXAMPLE 10.17: Split and merge.

Figure 10.43(a) shows a simple image. We define $P(R_i) = \text{TRUE}$ if at least 80% of the pixels in R_i have the property $|z_j - m_i| \leq 2\sigma_i$, where z_j is the gray level of the jth pixel in R_i, m_i is the mean gray level of that region, and σ_i is the

a b c

FIGURE 10.43
(a) Original image. (b) Result of split and merge procedure.
(c) Result of thresholding (a).



standard deviation of the gray levels in R_i. If $P(R_i)$ = TRUE under this condition, the values of all the pixels in R_i were set equal to m_i. Splitting and merging was done using the algorithm outlined previously. The result of applying this technique to the image in Fig. 10.43(a) is shown in Fig. 10.43(b). Note that the image was segmented perfectly. The image shown in Fig. 10.43(c) was obtained by thresholding Fig. 10.43(a), with a threshold placed midway between the two principal peaks of the histogram. The shading (and the stem of the leaf) were erroneously eliminated by the thresholding procedure. ■

As used in the preceding example, properties based on the mean and standard deviation of pixels in a region attempt to quantify the *texture* of a region (see Section 11.3.3 for a discussion on texture). The concept of *texture segmentation* is based on using measures of texture for the predicates $P(R_i)$. That is, we can perform texture segmentation by any of the methods discussed in this section by specifying predicates based on texture content.

10.5 Segmentation by Morphological Watersheds

Thus far, we have discussed segmentation based on three principal concepts: (a) detection of discontinuities, (b) thresholding, and (c) region processing. Each of these approaches was found to have advantages (for example, speed in the case of global thresholding) and disadvantages (for example, the need for post-processing, such as edge linking, in methods based on detecting discontinuities in gray levels). In this section we discuss an approach based on the concept of so-called *morphological watersheds*. As will become evident in the following discussion, segmentation by watersheds embodies many of the concepts of the other three approaches and, as such, often produces more stable segmentation results, including continuous segmentation boundaries. This approach also provides a simple framework for incorporating knowledge-based constraints (see Fig. 1.23) in the segmentation process.

10.5.1 Basic Concepts

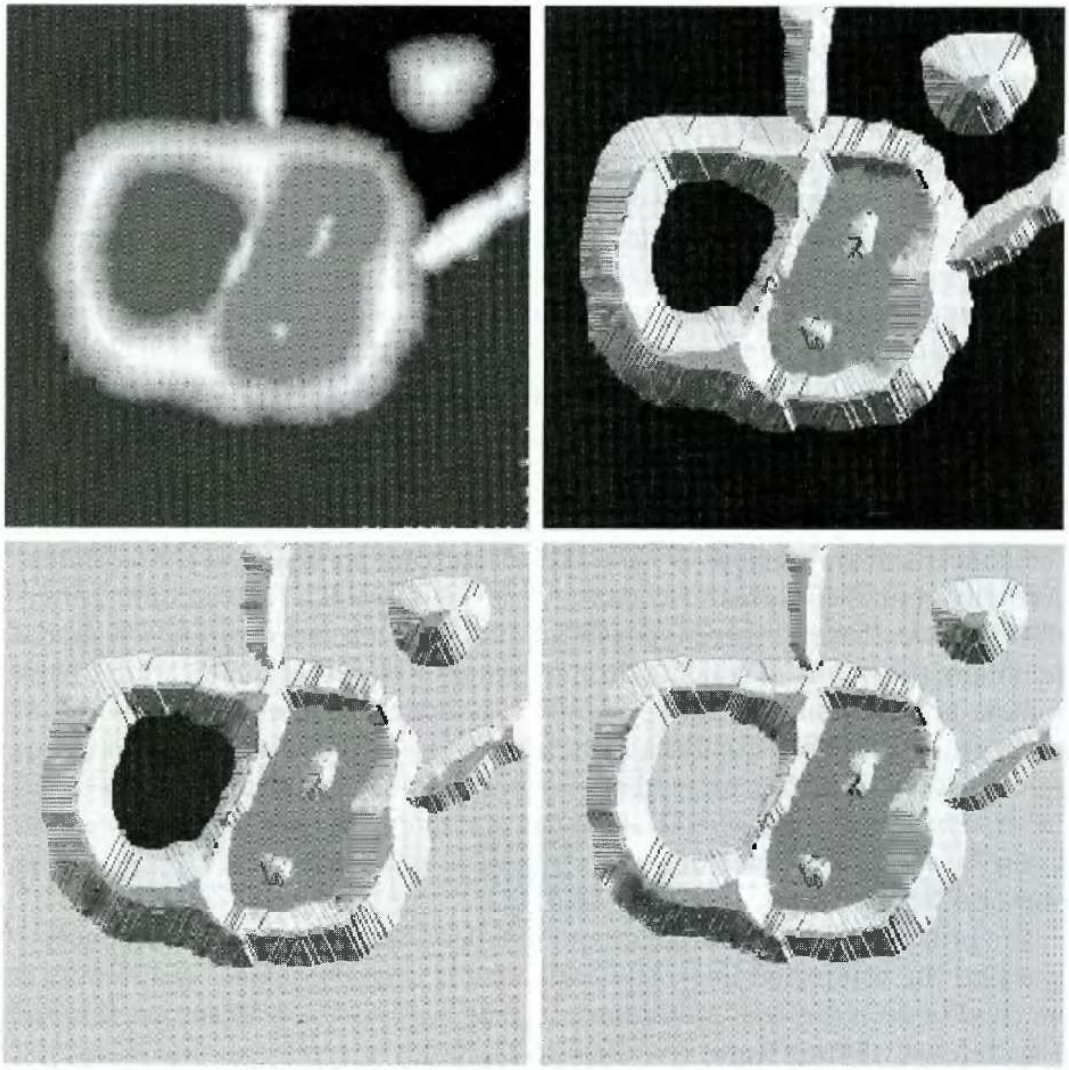
The concept of watersheds is based on visualizing an image in three dimensions: two spatial coordinates versus gray levels. In such a "topographic" interpretation, we consider three types of points: (a) points belonging to a regional minimum; (b) points at which a drop of water, if placed at the location of any of those points, would fall with certainty to a single minimum; and (c) points at which water would be equally likely to fall to more than one such minimum. For a particular regional minimum, the set of points satisfying condition (b) is called the *catchment basin* or *watershed* of that minimum. The points satisfying condition (c) form crest lines on the topographic surface and are termed *divide lines* or *watershed lines*.

The principal objective of segmentation algorithms based on these concepts is to find the watershed lines. The basic idea is simple: Suppose that a hole is punched in each regional minimum and that the entire topography is flooded from below by letting water rise through the holes at a uniform rate. When the

rising water in distinct catchment basins is about to merge, a dam is built to prevent the merging. The flooding will eventually reach a stage when only the tops of the dams are visible above the water line. These dam boundaries correspond to the divide lines of the watersheds. Therefore, they are the (continuous) boundaries extracted by a watershed segmentation algorithm.

These ideas can be explained further with the aid of Fig. 10.44. Figure 10.44(a) shows a simple gray-scale image and Fig. 10.44(b) is a topographic view, in which the height of the "mountains" is proportional to gray-level values in the input image. For ease of interpretation, the backsides of structures are shaded. This is not to be confused with gray-level values; only the general topography of the three-dimensional representation is of interest. In order to prevent the rising water from spilling out through the edges of the structure, we imagine the perimeter of the entire topography (image) being enclosed by dams of height greater than the highest possible mountain, whose value is determined by the highest possible gray-level value in the input image.

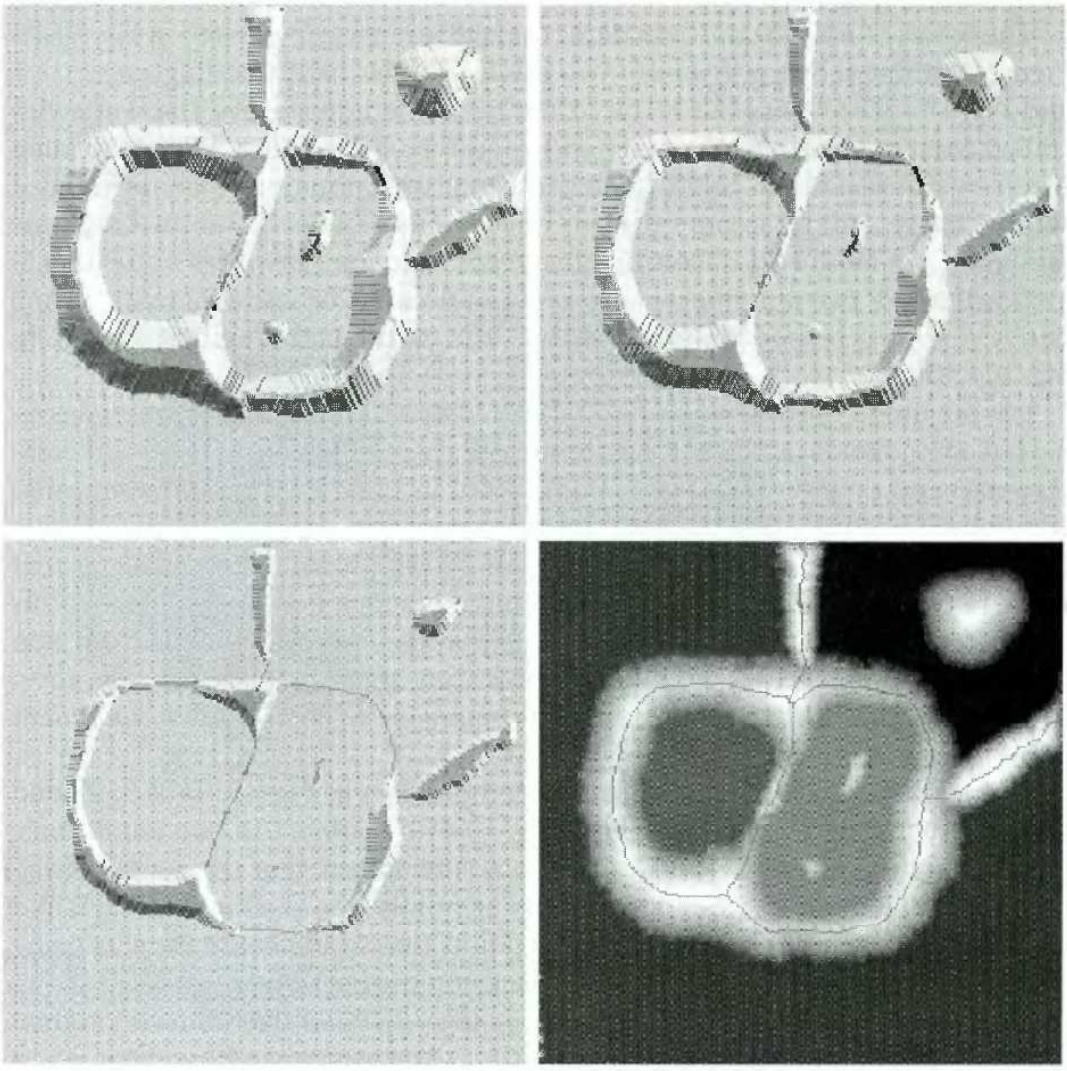
Suppose that a hole is punched in each regional minimum [shown as dark areas in Fig. 10.44(b)] and that the entire topography is flooded from below



a b
c d

FIGURE 10.44 (a) Original image. (b) Topographic view. (c)–(d) Two stages of flooding.

by letting water rise through the holes at a uniform rate. Figure 10.44(c) shows the first stage of flooding, where the "water," shown in light gray, has covered only areas that correspond to the very dark background in the image. In Figs. 10.44(d) and (e) we see that the water now has risen into the first and second catchment basins, respectively. As the water continues to rise, it will eventually overflow from one catchment basin into another. The first indication of this is shown in 10.44(f). Here, water from the left basin actually overflowed into the basin on the right and a short "dam" (consisting of single pixels) was built to prevent water from merging at that level of flooding (the details of dam building are discussed in the following section). The effect is more pronounced as water continues to rise, as shown in Fig. 10.44(g). This figure shows a longer dam between the two catchment basins and another dam in the top part of the right basin. The latter dam was built to prevent merging of water from that basin with water from areas corresponding to the background. This process is continued until the maximum level of flooding (corresponding to the highest gray-level value in the image) is reached. The final dams correspond to the watershed lines, which are the desired segmentation result. The



e f
g h

FIGURE 10.44 *(Continued)* (e) Result of further flooding. (f) Beginning of merging of water from two catchment basins (a short dam was built between them). (g) Longer dams. (h) Final watershed (segmentation) lines. (Courtesy of Dr. S. Beucher, CMM/Ecole des Mines de Paris.)

result for this example is shown in Fig. 10.44(h) as a dark, one-pixel-thick path superimposed on the original image. Note the important property that the watershed lines form a connected path, thus giving continuous boundaries between regions.

One of the principal applications of watershed segmentation is in the extraction of nearly uniform (bloblike) objects from the background. Regions characterized by small variations in gray levels have small gradient values. Thus, in practice, we often see watershed segmentation applied to the gradient of an image, rather than to the image itself. In this formulation, the regional minima of catchment basins correlate nicely with the small value of the gradient corresponding to the objects of interest.

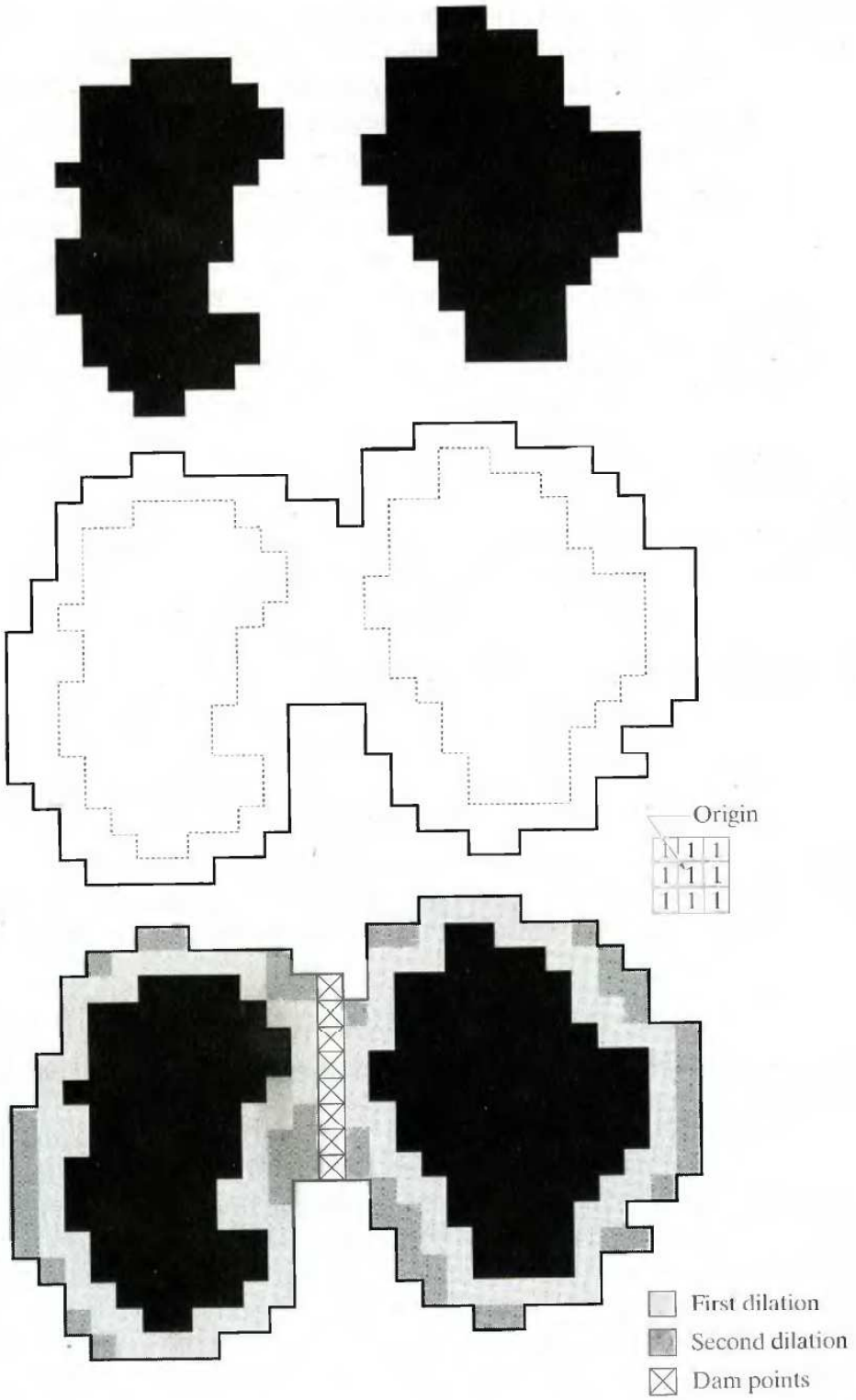
10.5.2 Dam Construction

Before proceeding, let us consider how to construct the dams or watershed lines required by watershed segmentation algorithms. Dam construction is based on binary images, which are members of 2-D integer space Z^2 (see Section 2.4.2). The simplest way to construct dams separating sets of binary points is to use morphological dilation (see Section 9.2.1).

The basics of how to construct dams using dilation are illustrated in Fig. 10.45. Figure 10.45(a) shows portions of two catchment basins at flooding step $n - 1$ and Fig. 10.45(b) shows the result at the next flooding step, n. The water has spilled from one basin to the other and, therefore, a dam must be built to keep this from happening. In order to be consistent with notation to be introduced shortly, let M_1 and M_2 denote the sets of coordinates of points in two regional minima. Then let the set of coordinates of points in the *catchment basin* associated with these two minima at stage $n - 1$ of flooding be denoted by $C_{n-1}(M_1)$ and $C_{n-1}(M_2)$, respectively. These are the two black regions shown in Fig. 10.45(a).

Let the union of these two sets be denoted by $C[n - 1]$. There are two connected components in Fig. 10.45(a) (see Section 2.5.2 regarding connected components) and only one connected component in Fig. 10.45(b). This connected component encompasses the earlier two components, shown dashed. The fact that two connected components have become a *single* component indicates that water between the two catchment basins has merged at flooding step n. Let this connected component be denoted q. Note that the two components from step $n - 1$ can be extracted from q by performing the simple AND operation $q \cap C[n - 1]$. We note also that all points belonging to an individual catchment basin form a single connected component.

Suppose that each of the connected components in Fig. 10.45(a) is dilated by the structuring element shown in Fig. 10.45(c), subject to two conditions: (1) The dilation has to be constrained to q (this means that the center of the structuring element can be located only at points in q during dilation), and (2) the dilation cannot be performed on points that would cause the sets being dilated to merge (become a single connected component). Figure 10.45(d) shows that a first dilation pass (in light gray) expanded the boundary of each original connected component. Note that condition (1) was satisfied by every point



a
b
d c

FIGURE 10.45 (a) Two partially flooded catchment basins at stage $n - 1$ of flooding. (b) Flooding at stage n, showing that water has spilled between basins (for clarity, water is shown in white rather than black). (c) Structuring element used for dilation. (d) Result of dilation and dam construction.

during dilation, and condition (2) did not apply to any point during the dilation process; thus the boundary of each region was expanded uniformly.

In the second dilation (shown in medium gray), several points failed condition (1) while meeting condition (2), resulting in the broken perimeter shown in the figure. It also is evident that the only points in q that satisfy the two conditions under consideration describe the one-pixel-thick connected path shown crossed-hatched in Fig. 10.45(d). This path constitutes the desired separating dam at stage n of flooding. Construction of the dam at this level of flooding is completed by setting all the points in the path just determined to a value greater than the maximum gray-level value of the image. The height of all dams is generally set at 1 plus the maximum allowed value in the image. This will prevent water from crossing over the part of the completed dam as the level of flooding is increased. It is important to note that dams built by this procedure, which are the desired segmentation boundaries, are connected components. In other words, this method eliminates the problems of broken segmentation lines.

Although the procedure just described is based on a simple example, the method used for more complex situations is exactly the same, including the use of the 3×3 symmetric structuring element shown in Fig. 10.45(c).

10.5.3 Watershed Segmentation Algorithm

Let $M_1, M_2, \ldots, M_R$ be sets denoting the *coordinates* of the points in the regional minima of an image $g(x, y)$. As indicated at the end of Section 10.5.1, this typically will be a gradient image. Let $C(M_i)$ be a set denoting the coordinates of the points in the catchment basin associated with regional minimum M_i (recall that the points in any catchment basin form a connected component). The notation min and max will be used to denote the minimum and maximum values of $g(x, y)$. Finally, let $T[n]$ represent the set of coordinates (s, t) for which $g(s, t) < n$. That is,

$$T[n] = \{(s, t) \mid g(s, t) < n\}. \tag{10.5-1}$$

Geometrically, $T[n]$ is the set of coordinates of points in $g(x, y)$ lying below the plane $g(x, y) = n$.

The topography will be flooded in *integer* flood increments, from $n = \min + 1$ to $n = \max + 1$. At any step n of the flooding process, the algorithm needs to know the number of points below the flood depth. Conceptually, suppose that the coordinates in $T[n]$ that are below the plane $g(x, y) = n$ are "marked" black, and all other coordinates are marked white. Then when we look "down" on the xy-plane at any increment n of flooding, we will see a binary image in which black points correspond to points in the function that are below the plane $g(x, y) = n$. This interpretation is quite useful in helping understand the following discussion.

Let $C_n(M_i)$ denote the set of coordinates of points in the catchment basin associated with minimum M_i that are flooded at stage n. With reference to the discussion in the previous paragraph, $C_n(M_i)$ may be viewed as a binary image given by

$$C_n(M_i) = C(M_i) \cap T[n]. \tag{10.5-2}$$

In other words, $C_n(M_i) = 1$ at location (x, y) if $(x, y) \in C(M_i)$ AND $(x, y) \in T[n]$; otherwise $C_n(M_i) = 0$. The geometrical interpretation of this result is straightforward. We are simply using the AND operator to isolate at stage n of flooding the portion of the binary image in $T[n]$ that is associated with regional minimum M_i.

Next, we let $C[n]$ denote the union of the flooded catchment basins portions at stage n:

$$C[n] = \bigcup_{i=1}^{R} C_n(M_i). \tag{10.5-3}$$

Then $C[\max + 1]$ is the union of all catchment basins:

$$C[\max + 1] = \bigcup_{i=1}^{R} C(M_i). \tag{10.5-4}$$

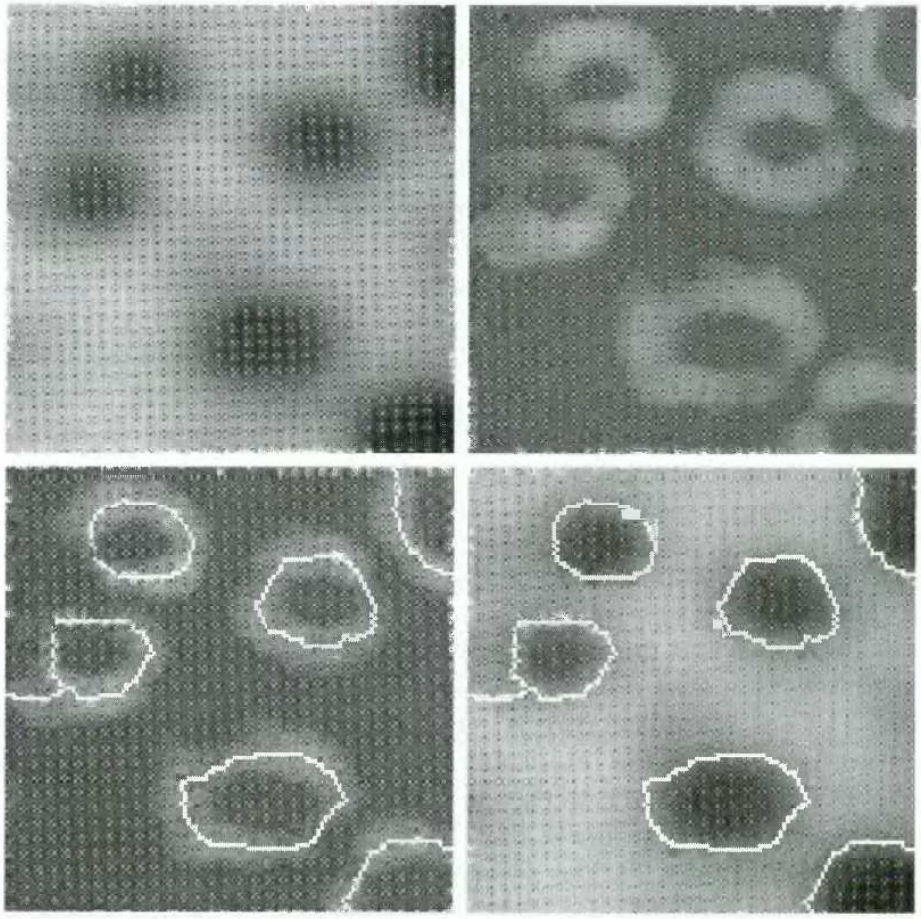
It can be shown (Problem 10.29) that the elements in both $C_n(M_i)$ and $T[n]$ are never replaced during execution of the algorithm, and that the number of elements in these two sets either increases or remains the same as n increases. Thus, it follows that $C[n-1]$ is a subset of $C[n]$. According to Eqs. (10.5-2) and (10.5-3), $C[n]$ is a subset of $T[n]$, so it follows that $C[n-1]$ is a subset of $T[n]$. From this we have the important result that each connected component of $C[n-1]$ is contained in exactly one connected component of $T[n]$.

The algorithm for finding the watershed lines is initialized with $C[\min + 1] = T[\min + 1]$. The algorithm then proceeds recursively, assuming at step n that $C[n-1]$ has been constructed. A procedure for obtaining $C[n]$ from $C[n-1]$ is as follows. Let Q denote the set of connected components in $T[n]$. Then, for each connected component $q \in Q[n]$, there are three possibilities:

(a) $q \cap C[n-1]$ is empty.
(b) $q \cap C[n-1]$ contains one connected component of $C[n-1]$.
(c) $q \cap C[n-1]$ contains more than one connected component of $C[n-1]$.

Construction of $C[n]$ from $C[n-1]$ depends on which of these three conditions holds. Condition (a) occurs when a new minimum is encountered, in which case connected component q is incorporated into $C[n-1]$ to form $C[n]$. Condition (b) occurs when q lies within the catchment basin of some regional minimum, in which case q is incorporated into $C[n-1]$ to form $C[n]$. Condition (c) occurs when all, or part, of a ridge separating two or more catchment basins is encountered. Further flooding would cause the water level in these catchment basins to merge. Thus a dam (or dams if more than two catchment basins are involved) must be built within q to prevent overflow between the catchment basins. As explained in the previous section, a one-pixel-thick dam can be constructed when needed by dilating $q \cap C[n-1]$ with a 3×3 structuring element of 1's, and constraining the dilation to q.

Algorithm efficiency is improved by using only values of n that correspond to existing gray-level values in $g(x, y)$; we can determine these values, as well as the values of min and max, from the histogram of $g(x, y)$.



a b
c d

FIGURE 10.46 (a) Image of blobs. (b) Image gradient. (c) Watershed lines. (d) Watershed lines superimposed on original image. (Courtesy of Dr. S. Beucher, CMM/Ecole des Mines de Paris.)

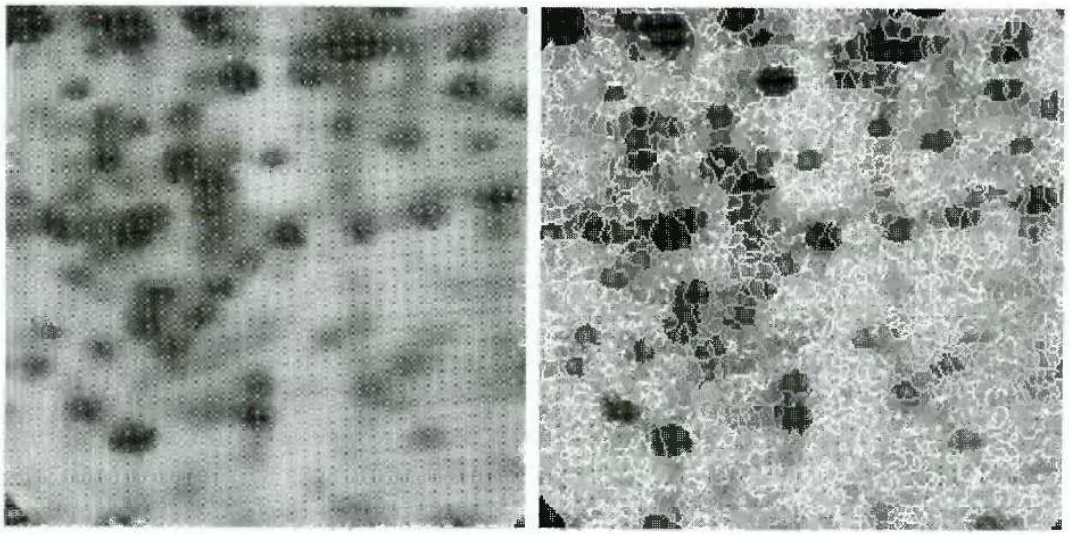
EXAMPLE 10.18: Illustration of the watershed segmentation algorithm.

■ Consider the image and its gradient, shown in Figs. 10.46(a) and (b), respectively. Application of the watershed algorithm just described yielded the watershed lines (white paths) of the gradient image shown in Fig. 10.46(c). These segmentation boundaries are shown superimposed on the original image in Fig. 10.46(d). As noted at the beginning of this section, the segmentation boundaries have the important property of being connected paths. ■

10.5.4 The Use of Markers

Direct application of the watershed segmentation algorithm in the form discussed in the previous section generally leads to *oversegmentation* due to noise and other local irregularities of the gradient. As shown in Fig. 10.47, oversegmentation can be serious enough to render the result of the algorithm virtually useless. In this case, this means a large number of segmented regions. A practical solution to this problem is to limit the number of allowable regions by incorporating a preprocessing stage designed to bring additional knowledge into the segmentation procedure.

An approach used to control oversegmentation is based on the concept of markers. A *marker* is a connected component belonging to an image. We have *internal* markers, associated with objects of interest, and *external* markers, associated with the background. A procedure for marker selection typically will

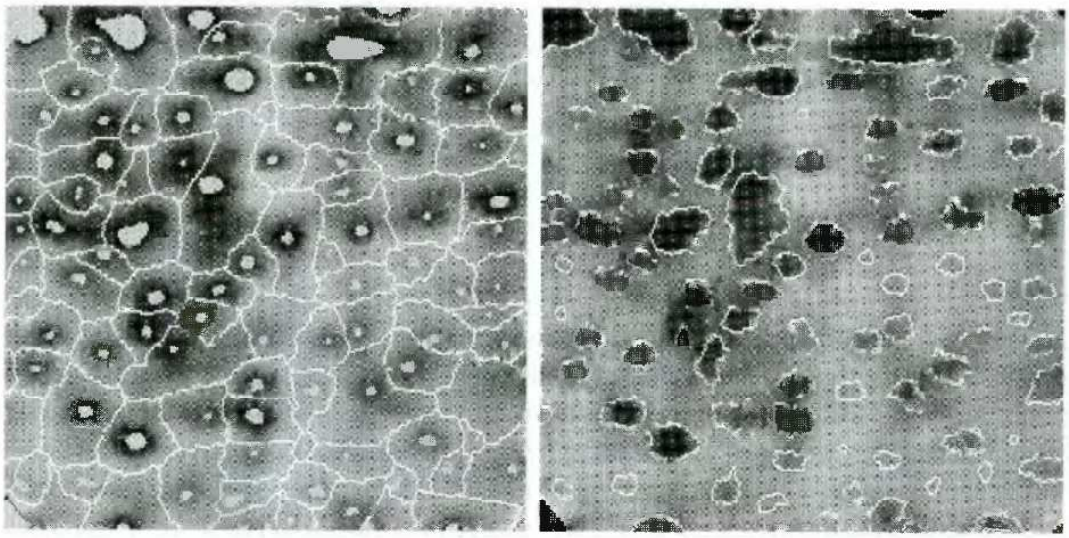


a b

FIGURE 10.47 (a) Electrophoresis image. (b) Result of applying the watershed segmentation algorithm to the gradient image. Oversegmentation is evident. (Courtesy of Dr. S. Beucher, CMM/Ecole des Mines de Paris.)

consist of two principal steps: (1) preprocessing; and (2) definition of a set of criteria that markers must satisfy. To illustrate, consider Fig. 10.47(a) again. Part of the problem that led to the oversegmented result in Fig. 10.47(b) is the large number of potential minima. Because of their size, many of these minima really are irrelevant detail. As has been pointed out several times in earlier discussions, an effective method for minimizing the effect of small spatial detail is to filter the image with a smoothing filter. This is an appropriate preprocessing scheme in this particular case.

Suppose that we define an internal marker in this case as (1) a region that is surrounded by points of higher "altitude;" (2) such that the points in the region form a connected component; and (3) in which all the points in the connected component have the same gray-level value. After the image was smoothed, the internal markers resulting from this definition are shown as light gray, bloblike regions in Fig. 10.48(a). Next, the watershed algorithm was applied to the smoothed



a b

FIGURE 10.48 (a) Image showing internal markers (light gray regions) and external markers (watershed lines). (b) Result of segmentation. Note the improvement over Fig. 10.47(b). (Courtesy of Dr. S. Beucher, CMM/Ecole des Mines de Paris.)

image, under the restriction that these internal markers be the *only* allowed regional minima. Figure 10.48(a) shows the resulting watershed lines. These watershed lines are defined as the external markers. Note that the points along the watershed lines are good candidates for the background because they pass along the highest points between neighboring markers.

The external markers shown in Fig. 10.48(a) effectively partition the image into regions, with each region containing a single internal marker and part of the background. The problem is thus reduced to partitioning each of these regions into two: a single object and its background. We can bring to bear on this simplified problem many of the segmentation techniques discussed earlier in this chapter. Another approach is simply to apply the watershed segmentation algorithm to each individual region. In other words, we simply take the gradient of the smoothed image [as in Fig. 10.46(b)] and then restrict the algorithm to operate on a single watershed that contains the marker in that particular region. The result obtained using this approach is shown in 10.48(b). The improvement over the image in 10.47(b) is evident.

Marker selection can range from simple procedures based on gray-level values and connectivity, as was just illustrated, to more complex descriptions involving size, shape, location, relative distances, texture content, and so on (see Chapter 11 regarding descriptors). The point is that using markers brings a priori knowledge to bear on the segmentation problem. The reader is reminded that humans often aid segmentation and higher-level tasks in every-day vision by using a priori knowledge, one of the most familiar being the use of context. Thus, the fact that segmentation by watersheds offers a framework that can make effective use of this type of knowledge is a significant advantage of this method.

10.6 The Use of Motion in Segmentation

Motion is a powerful cue used by humans and many animals to extract objects of interest from a background of irrelevant detail. In imaging applications, motion arises from a relative displacement between the sensing system and the scene being viewed, such as in robotic applications, autonomous navigation, and dynamic scene analysis. In the following sections we consider the use of motion in segmentation both spatially and in the frequency domain.

10.6.1 Spatial Techniques

Basic approach

One of the simplest approaches for detecting changes between two image frames $f(x, y, t_i)$ and $f(x, y, t_j)$ taken at times t_i and t_j, respectively, is to compare the two images pixel by pixel. One procedure for doing this is to form a difference image. Suppose that we have a reference image containing only stationary components. Comparing this image against a subsequent image of the same scene, but including a moving object, results in the difference of the two images canceling the stationary elements, leaving only nonzero entries that correspond to the nonstationary image components.

A difference image between two images taken at times t_i and t_j may be defined as

$$d_{ij}(x, y) = \begin{cases} 1 & \text{if } |f(x, y, t_i) - f(x, y, t_j)| > T \\ 0 & \text{otherwise} \end{cases} \tag{10.6-1}$$

where T is a specified threshold. Note that $d_{ij}(x, y)$ has a value of 1 at spatial coordinates (x, y) only if the gray-level difference between the two images is appreciably different at those coordinates, as determined by the specified threshold T. It is assumed that all images are of the same size. Finally, we note that the values of the coordinates (x, y) in Eq. (10.6-1) span the dimensions of these images, so that the difference image $d_{ij}(x, y)$ also is of same size as the images in the sequence.

In dynamic image processing, all pixels in $d_{ij}(x, y)$ with value 1 are considered the result of object motion. This approach is applicable only if the two images are registered spatially and if the illumination is relatively constant within the bounds established by T. In practice, 1-valued entries in $d_{ij}(x, y)$ often arise as a result of noise. Typically, these entries are isolated points in the difference image, and a simple approach to their removal is to form 4- or 8-connected regions of 1's in $d_{ij}(x, y)$ and then ignore any region that has less than a predetermined number of entries. Although it may result in ignoring small and/or slow-moving objects, this approach improves the chances that the remaining entries in the difference image actually are the result of motion.

Accumulative differences

Isolated entries resulting from noise is not an insignificant problem when trying to extract motion components from a sequence of images. Although the number of these entries can be reduced by a thresholded connectivity analysis, this filtering process can also remove small or slow-moving objects as noted in the previous section. One way to address this problem is by considering changes at a pixel location over several frames, thus introducing a "memory" into the process. The idea is to ignore changes that occur only sporadically over a frame sequence and can therefore be attributed to random noise.

Consider a sequence of image frames $f(x, y, t_1), f(x, y, t_2), \ldots, f(x, y, t_n)$ and let $f(x, y, t_1)$ be the reference image. An *accumulative difference image* (ADI) is formed by comparing this reference image with every subsequent image in the sequence. A counter for each pixel location in the accumulative image is incremented every time a difference occurs at that pixel location between the reference and an image in the sequence. Thus when the kth frame is being compared with the reference, the entry in a given pixel of the accumulative image gives the number of times the gray level at that position was different from the corresponding pixel value in the reference image. Differences are established, for example, by using Eq. (10.6-1).

Often useful is consideration of three types of accumulative difference images: *absolute*, *positive*, and *negative* ADIs. Assuming that the gray-level values of the moving objects are larger than the background, these three types of ADIs are defined as follows. Let $R(x, y)$ denote the reference image and, to simplify

the notation, let k denote t_k, so that $f(x, y, k) = f(x, y, t_k)$. We assume that $R(x, y) = f(x, y, 1)$. Then, for any $k > 1$, and keeping in mind that the values of the ADIs are *counts*, we define the following for all relevant values of (x, y):

$$A_k(x, y) = \begin{cases} A_{k-1}(x, y) + 1 & \text{if } |R(x, y) - f(x, y, k)| > T \\ A_{k-1}(x, y) & \text{otherwise} \end{cases} \tag{10.6-2}$$

$$P_k(x, y) = \begin{cases} P_{k-1}(x, y) + 1 & \text{if } [R(x, y) - f(x, y, k)] > T \\ P_{k-1}(x, y) & \text{otherwise} \end{cases} \tag{10.6-3}$$

and

$$N_k(x, y) = \begin{cases} N_{k-1}(x, y) + 1 & \text{if } [R(x, y) - f(x, y, k)] < -T \\ N_{k-1}(x, y) & \text{otherwise} \end{cases} \tag{10.6-4}$$

where $A_k(x, t)$, $P_k(x, y)$, and $N_k(x, y)$ are the absolute, positive, and negative ADIs, respectively, after the kth image in the sequence is encountered.

It is understood that these ADIs start out with all zero values (counts). Note also that the ADIs are the same size as the images in the sequence. As noted previously, the images in the sequence are all assumed to be of the same size. Finally, we note that the order of the inequalities and signs of the thresholds in Eqs. (10.6-3) and (10.6-4) are reversed if the gray-level values of the background pixels are greater than the levels of the moving objects.

EXAMPLE 10.19: Computation of the absolute, positive, and negative accumulative difference images.

■ Figure 10.49 shows the three ADIs displayed as intensity images for a rectangular object of dimension 75×50 pixels that is moving in a southeasterly direction at a speed of $5\sqrt{2}$ pixels per frame. The images are of size 256×256 pixels. We note the following: (1) The nonzero area of the positive ADI is equal to the size of the moving object. (2) The location of the positive ADI corresponds to the location of the moving object in the reference frame. (3) The number of counts in the positive ADI stops increasing when the moving object is displaced completely with respect to the same object in the reference frame.

a b c

FIGURE 10.49 ADIs of a rectangular object moving in a southeasterly direction. (a) Absolute ADI. (b) Positive ADI. (c) Negative ADI.

(4) The absolute ADI contains the regions of the positive and negative ADI.
(5) The direction and speed of the moving object can be determined from the entries in the absolute and negative ADIs. ■

Establishing a reference image

A key to the success of the techniques discussed in the preceding two sections is having a reference image against which subsequent comparisons can be made. As indicated, the difference between two images in a dynamic imaging problem has the tendency to cancel all stationary components, leaving only image elements that correspond to noise and to the moving objects. The noise problem can be handled by the filtering approach mentioned earlier or by forming an accumulative difference image, as discussed in the preceding section.

In practice, obtaining a reference image with only stationary elements is not always possible, and building a reference from a set of images containing one or more moving objects becomes necessary. This necessity applies particularly to situations describing busy scenes or in cases where frequent updating is required. One procedure for generating a reference image is as follows. Consider the first image in a sequence to be the reference image. When a nonstationary component has moved completely out of its position in the reference frame, the corresponding background in the present frame can be duplicated in the location originally occupied by the object in the reference frame. When all moving objects have moved completely out of their original positions, a reference image containing only stationary components will have been created. Object displacement can be established by monitoring the changes in the positive ADI, as indicated in the preceding section.

EXAMPLE 10.20: Building a reference image.

■ Figures 10.50(a) and (b) show two image frames of a traffic intersection. The first image is considered the reference, and the second depicts the same scene some time later. The objective is to remove the principal moving objects in the reference image in order to create a static image. Although there are other smaller moving objects, the principal moving feature is the automobile at the intersection moving from left to right. For illustrative purposes we focus on this object. By monitoring the changes in the positive ADI, it is possible to determine the initial position of a moving object, as explained previously. Once the area

a b c

FIGURE 10.50 Building a static reference image. (a) and (b) Two frames in a sequence. (c) Eastbound automobile subtracted from (a) and the background restored from the corresponding area in (b). (Jain and Jain.)

occupied by this object is identified, the object can be removed from the image by subtraction. By looking at the frame in the sequence at which the positive ADI stopped changing, we can copy from this image the area previously occupied by the moving object in the initial frame. This area then is pasted onto the image from which the object was cut out, thus restoring the background of that area. If this is done for all moving objects, the result is a reference image with only static components against which we can compare subsequent frames for motion detection, as explained in the previous two sections. The result of removing the east-bound moving vehicle in this case is shown in Fig. 10.50(c). ■

10.6.2 Frequency Domain Techniques

In this section we consider the problem of determining motion estimates via a Fourier transform formulation. Consider a sequence $f(x, y, t)$, $t = 0, 1, \ldots, K - 1$, of K digital image frames of size $M \times N$ generated by a stationary camera. We begin the development by assuming that all frames have a homogeneous background of zero intensity. The exception is a single, 1-pixel object of unit intensity that is moving with constant velocity. Suppose that for frame one ($t = 0$) the image plane is *projected* onto the x-axis; that is, the pixel intensities are summed across the columns. This operation yields a 1-D array with M entries that are 0, except at the location where the object is projected. Multiplying the components of the array by $\exp[j2\pi a_1 x\Delta t]$, $x = 0, 1, 2, \ldots, M - 1$, with the object at coordinates (x', y') at that instant of time, produces a sum equal to $\exp[j2\pi a_1 x'\Delta t]$. In this notation a_1 is a positive integer, and Δt is the time interval between frames.

Suppose that in frame two ($t = 1$) the object has moved to coordinates $(x' + 1, y')$; that is, it has moved 1 pixel parallel to the x-axis. Then repeating the projection procedure discussed in the previous paragraph yields the sum $\exp[j2\pi a_1(x' + 1)\Delta t]$. If the object continues to move 1 pixel location per frame, then, at any integer instant of time, the result is $\exp[j2\pi a_1(x' + t)\Delta t]$, which, using Euler's formula, may be expressed as

$$e^{j2\pi a_1(x'+t)\Delta t} = \cos[2\pi a_1(x' + t)\Delta t] + j\sin[2\pi a_1(x' + t)\Delta t] \quad (10.6\text{-}5)$$

for $t = 0, 1, \ldots, K - 1$. In other words, this procedure yields a complex sinusoid with frequency a_1. If the object were moving v_1 pixels (in the x-direction) between frames, the sinusoid would have frequency $v_1 a_1$. Because t varies between 0 and $K - 1$ in integer increments, restricting a_1 to integer values causes the discrete Fourier transform of the complex sinusoid to have two peaks—one located at frequency $v_1 a_1$ and the other at $K - v_1 a_1$. This latter peak is the result of symmetry in the discrete Fourier transform, as discussed in Section 4.6, and may be ignored. Thus a peak search in the Fourier spectrum yields $v_1 a_1$. Division of this quantity by a_1 yields v_1, which is the velocity component in the x-direction, as the frame rate is assumed to be known. A similar argument would yield v_2, the component of velocity in the y-direction.

A sequence of frames in which no motion takes place produces identical exponential terms, whose Fourier transform would consist of a single peak at a frequency of 0 (a single dc term). Therefore, because the operations discussed

so far are linear, the general case involving one or more moving objects in an arbitrary static background would have a Fourier transform with a peak at dc corresponding to static image components and peaks at locations proportional to the velocities of the objects.

These concepts may be summarized as follows. For a sequence of K digital images of size $M \times N$, the sum of the weighted projections onto the x axis at any integer instant of time is

$$g_x(t, a_1) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y, t) e^{j2\pi a_1 x \Delta t} \qquad t = 0, 1, \ldots, K-1. \tag{10.6-6}$$

Similarly, the sum of the projections onto the y-axis is

$$g_y(t, a_2) = \sum_{y=0}^{N-1} \sum_{x=0}^{M-1} f(x, y, t) e^{j2\pi a_2 y \Delta t} \qquad t = 0, 1, \ldots, K-1 \tag{10.6-7}$$

where, as noted already, a_1 and a_2 are positive integers.

The 1-D Fourier transforms of Eqs. (10.6-6) and (10.6-7), respectively, are

$$G_x(u_1, a_1) = \frac{1}{K} \sum_{t=0}^{K-1} g_x(t, a_1) e^{-j2\pi u_1 t/K} \qquad u_1 = 0, 1, \ldots, K-1 \tag{10.6-8}$$

and

$$G_y(u_2, a_2) = \frac{1}{K} \sum_{t=0}^{K-1} g_y(t, a_2) e^{-j2\pi u_2 t/K} \qquad u_2 = 0, 1, \ldots, K-1. \tag{10.6-9}$$

In practice, computation of these transforms is carried out using an FFT algorithm, as discussed in Section 4.6.

The frequency-velocity relationship is

$$u_1 = a_1 v_1 \tag{10.6-10}$$

and

$$u_2 = a_2 v_2. \tag{10.6-11}$$

In this formulation the unit of velocity is in pixels per total frame time. For example, $v_1 = 10$ is interpreted as a motion of 10 pixels in K frames. For frames that are taken uniformly, the actual physical speed depends on the frame rate and the distance between pixels. Thus if $v_1 = 10$, $K = 30$, the frame rate is two images per second, and the distance between pixels is 0.5 m, then the actual physical speed in the x-direction is

$$\begin{aligned} v_1 &= (10 \text{ pixels})(0.5 \text{ m/pixel})(2 \text{ frames/s})/(30 \text{ frames}) \\ &= 1/3 \text{m/s}. \end{aligned}$$

The sign of the x-component of the velocity is obtained by computing

$$S_{1x} = \left. \frac{d^2 \mathrm{Re}[g_x(t, a_1)]}{dt^2} \right|_{t=n} \tag{10.6-12}$$

and

$$S_{2x} = \left.\frac{d^2 \mathrm{Im}\left[g_x(t, a_1)\right]}{dt^2}\right|_{t=n} \tag{10.6-13}$$

Because g_x is sinusoidal, it can be shown that S_{1x} and S_{2x} will have the same sign at an arbitrary point in time, n, if the velocity component v_1 is positive. Conversely, opposite signs in S_{1x} and S_{2x} indicate a negative component. If either S_{1x} or S_{2x} is zero, we consider the next closest point in time, $t = n \pm \Delta t$. Similar comments apply to computing the sign of v_2.

EXAMPLE 10.21: Detection of a small moving object via the frequency domain.

■ Figures 10.51 through 10.54 illustrate the effectiveness of the approach just derived. Figure 10.51 shows one of a 32-frame sequence of LANDSAT images generated by adding white noise to a reference image. The sequence contains a superimposed target moving at 0.5 pixel per frame in the x-direction and 1 pixel per frame in the y-direction. The target, shown circled in Fig. 10.52, has a Gaussian intensity distribution spread over a small (9-pixel) area and is not easily discernible by eye. The results of computing Eqs. (10.6-8) and (10.6-9) with $a_1 = 6$ and $a_2 = 4$ are shown in Figs. 10.53 and 10.54, respectively. The peak at $u_1 = 3$ in Fig. 10.53 yields $v_1 = 0.5$ from Eq. (10.6-10). Similarly, the peak at in Fig. 10.54 yields $v_2 = 1.0$ from Eq. (10.6-11). ■

Guidelines for the selection of a_1 and a_2 can be explained with the aid of Figs. 10.53 and 10.54. For instance, suppose that we had used $a_2 = 15$ instead of $a_2 = 4$. In that case the peaks in Fig. 10.54 would now be at $u_2 = 15$ and 17 because $v_2 = 1.0$, which would be a seriously aliased result. As discussed in Section 2.4.4, aliasing is caused by undersampling (too few frames in the present discussion, as the range of u is determined by K). Because $u = av$, one possibility is to select a as the integer closest to $a = u_{\max}/v_{\max}$, where $u_{\max}$ is the aliasing frequency limitation established K and $v_{\max}$ is the maximum expected object velocity.

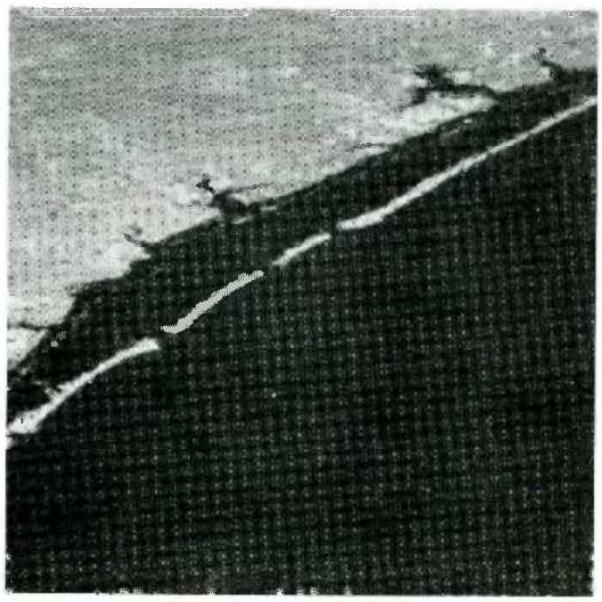


FIGURE 10.51 LANDSAT frame. (Cowart, Snyder, and Ruedger.)

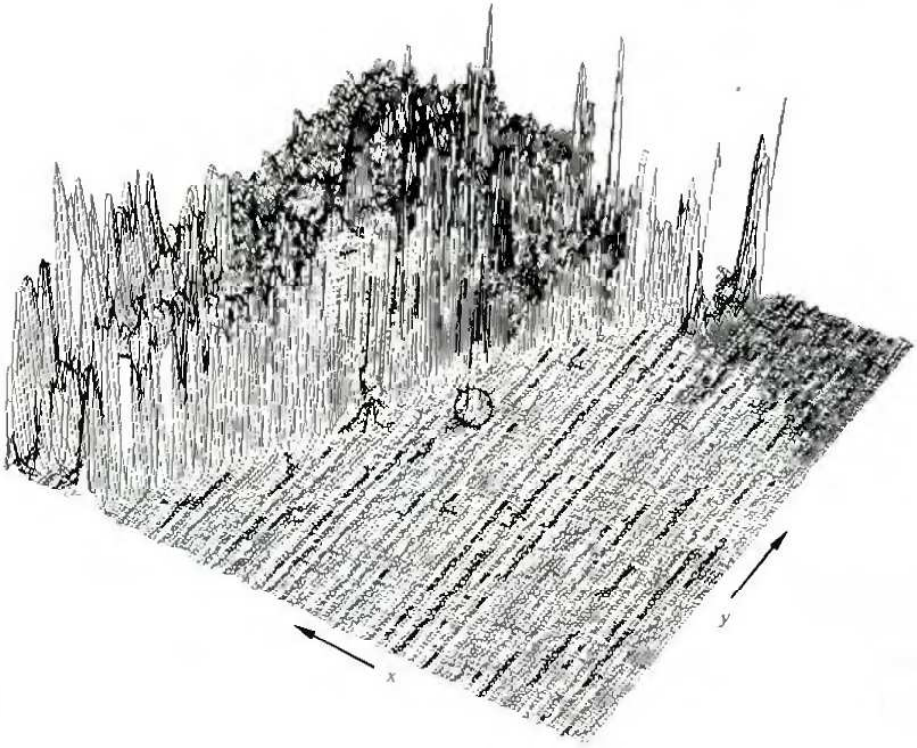


FIGURE 10.52 Intensity plot of the image in Fig. 10.51, with the target circled. (Rajala, Riddle, and Snyder.)

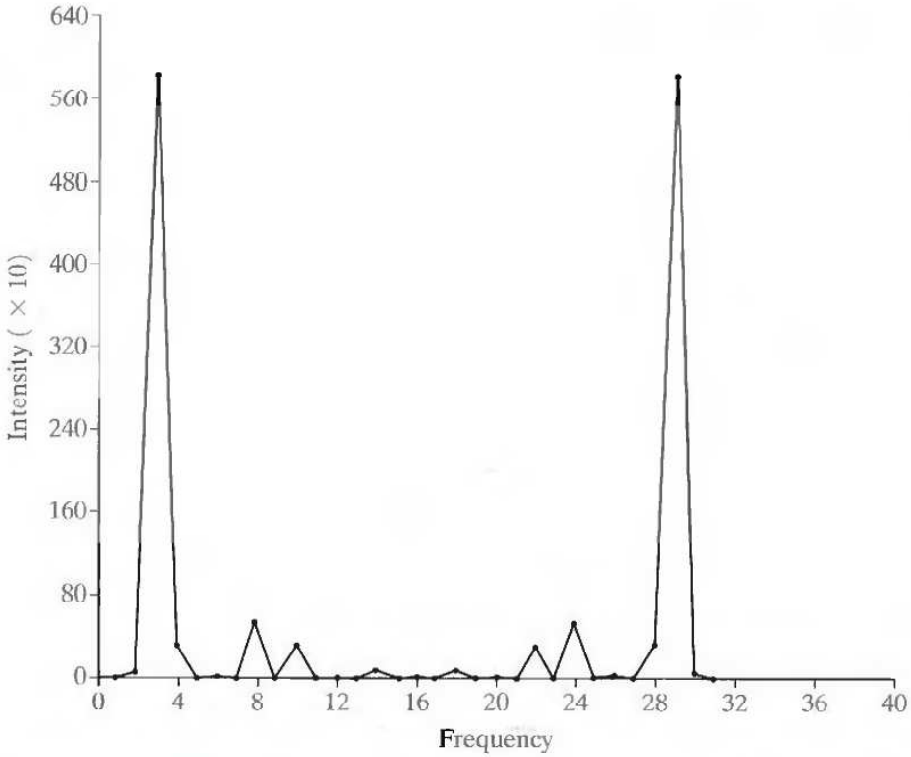


FIGURE 10.53 Spectrum of Eq. (10.6-8) showing a peak at $u_1 = 3$. (Rajala, Riddle, and Snyder.)

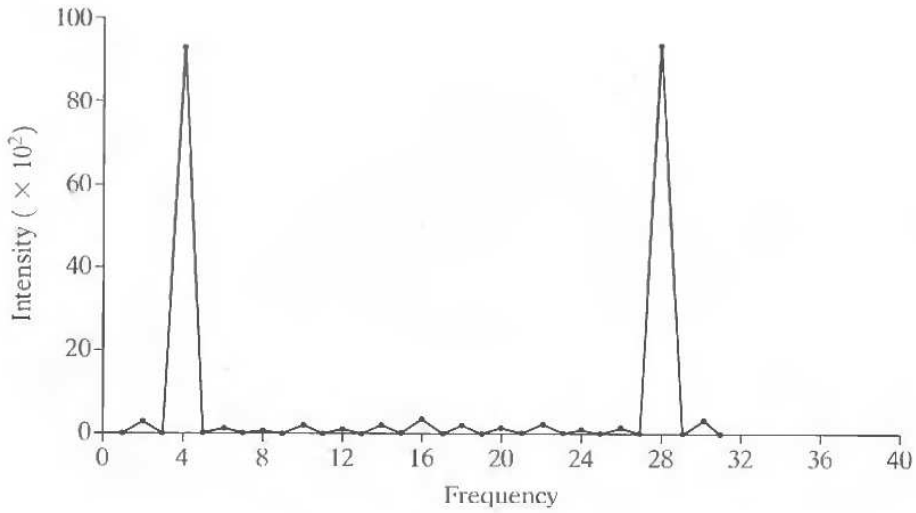


FIGURE 10.54 Spectrum of Eq. (10.6-9) showing a peak at $u_2 = 4$. (Rajala, Riddle, and Snyder.)

Summary

Image segmentation is an essential preliminary step in most automatic pictorial pattern recognition and scene analysis problems. As indicated by the range of examples presented in the previous sections, the choice of one segmentation technique over another is dictated mostly by the peculiar characteristics of the problem being considered. The methods discussed in this chapter, although far from exhaustive, are representative of techniques commonly used in practice. The following references can be used as the basis for further study of this topic.

References and Further Reading

Because of its central role in autonomous image processing, segmentation is a topic covered in most books dealing with image processing, image analysis, and computer vision. The following books provide complementary and/or supplementary reading for our coverage of this topic: Shapiro and Stockman [2001], Sonka et al. [1999], Petrou and Bosdogianni [1999], and Umbaugh [1998].

Work dealing with the use of masks to detect gray-level discontinuities (Section 10.1) has a long history. Numerous masks have been proposed over the years: Roberts [1965], Prewitt [1970], Kirsh [1971], Robinson [1976], Frei and Chen [1977], and Canny [1986]. A review article by Fram and Deutsch [1975] contains numerous masks and an evaluation of their performance. The issue of mask performance, especially for edge detection, still is an area of considerable interest, as exemplified by Qian and Huang [1996], Wang et al. [1996], Heath et al. [1997, 1998], and Ando [2000]. Edge detection on color images has been increasing in popularity for a number of multisensing applications. See, for example, Salinas, Abidi and Gonzalez [1996]; Zugaj and Lattuati [1998]; Mirmehdi and Petrou [2000]; and Plataniotis and Venetsanopoulos [2000]. The interplay between image characteristic and mask performance also is a topic of current interest, as exemplified by Ziou [2001]. Our presentation of the zero-crossing properties of the Laplacian is based on a paper by Marr and Hildredth [1980] and on the book by Marr [1982]. See also a paper by Clark [1989] on authenticating edges produced by zero-crossing algorithms. (Corrections of parts of the Clark paper are given by Piech [1990].) As mentioned in Section 10.1, zero crossing via the Laplacian of a Gaussian is an important approach whose relative performance is still an active topic of research (Gunn [1998, 1999]).

The Hough transform (Hough [1962]) has emerged over the past decade as a method of choice for global pixel linking and curve detection. Numerous generalizations to the basic transform discussed in this chapter have been proposed over the years. For example, Lo and Tsai [1995] discuss an approach for detecting thick lines, Guil et al. [1995, 1997] deal with fast implementations of the Hough transform and detection of primitive curves, Daul at al. [1998] discuss further generalizations for detecting elliptical arcs, and Shapiro [1996] deals with implementation of the Hough transform for gray-scale images. The algorithm presented in Section 10.2.3 is based on Martelli [1972, 1976]. For additional reading on heuristic graph searching, see Nilsson [1980], Umeyama [1988], and Sonka et al. [1999].

As mentioned at the beginning of Section 10.3, thresholding techniques enjoy a significant degree of popularity because they are simple to implement. It is not surprising that there is a considerable body of work reported in the literature on this topic. A good appreciation of the extent of this literature can be gained from the review papers by Sahoo et al. [1988] and by Lee et al. [1990]. We spent a significant level of effort in Section 10.3.2 dealing with the effects of illumination on thresholding. The types of approaches used to deal with this problem are illustrated by the work of Perez and Gonzalez [1987], Parker [1991], Murase and Nayar [1994], Bischsel [1998], and Drew et al. [1999]. For additional reading on the material in Sections 10.3.3 and 10.3.4, see Jain et al. [1995]. The early work of Chow and Kaneko [1972] discussed in Section 10.3.5 is still a standard in terms of illustrating the key aspects of a threshold-based image segmentation solution. Essentially the same can be said for the material presented in Section 10.3.6 (due to White and Rohrer [1983]), which combines thresholding, the gradient, and the Laplacian in the solution of a difficult segmentation problem. It is interesting to compare the fundamental similarities in terms of image segmentation capability between these two articles and work on thresholding done almost twenty years later (Cheriet et al. [1998], Sauvola and Pietikainen [2000]). See also Liang et al. [2000] and Chan et al. [2000] for alternate approaches to the problem of detecting boundaries in images similar in concept to those studied by Chow and Kaneko.

See Fu and Mui [1981] for an early survey on the topic of region-oriented segmentation. The works of Haddon and Boyce [1990] and of Pavlidis and Liow [1990] are among the earliest efforts to integrate region and boundary information for the purpose of segmentation. A newer region-growing approach proposed by Hojjatoleslami and Kittler [1998] also is of interest. For current basic coverage of region-oriented segmentation concepts, see Shapiro and Stockman [2001] and Sonka et al. [1999].

Segmentation by watersheds was shown in Section 10.5 to be a powerful concept. Early references dealing with segmentation by watersheds are Serra [1988], Beucher [1990], and Beucher and Meyer [1992]. The paper by Baccar et al. [1996] discusses segmentation based on data fusion and morphological watersheds. The progress in this field in a little more than one decade is evident in a special issue of *Pattern Recognition* [2000], devoted entirely to this topic. As indicated in our discussion in Section 10.5, one of the key issues with watersheds is the problem of over segmentation. The papers by Najmanand and Schmitt [1996], Haris et al. [1998], and Bleau and Leon [2000] are illustrative of approaches for dealing with this problem. Bieniek and Moga [2000] discuss a watershed segmentation algorithm based on connected components.

The material in Section 10.6.1 is from Jain, R. [1981]. See also Jain, Kasturi, and Schunck [1995]. The material in Section 10.6.2 is from Rajala, Riddle, and Snyder [1983]. See also the papers by Shariat and Price [1990] and by Cumani et al. [1991]. The books by Shapiro and Stockman [2001] and by Sonka et al. [1999] provide additional reading regarding motion estimation.

Problems

★**10.1** A binary image contains straight lines oriented horizontally, vertically, at 45°, and at −45°. Give a set of 3 × 3 masks that can be used to detect 1-pixel-long breaks in these lines. Assume that the gray level of the lines is 1 and that the gray level of the background is 0.

10.2 Propose a technique for detecting gaps of length ranging between 1 and L pixels in line segments of a binary image. Assume that the lines are 1 pixel thick. Base your technique on 8-neighbor connectivity analysis, rather than attempting to construct masks for detecting the gaps.

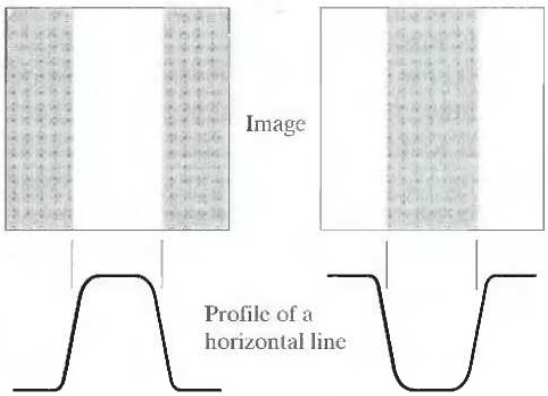
10.3 Refer to Fig. 10.4 in answering the following questions.

★ **(a)** Some of the lines joining the pads and center element in Fig. 10.4(b) are single lines, while some others are double lines. Explain why.

(b) How would you go about eliminating the components in Fig. 10.4(c) that are not part of the line oriented at −45°?

10.4 Consider a horizontal intensity profile through the middle of a binary image that contains a step edge running vertically through the center of the image. Draw what the profile would look like after the image has been blurred by an averaging mask of size $n \times n$, with coefficients equal to $1/n^2$. For simplicity, assume that the image was scaled so that its gray levels are 0 on the left of the edge and 1 to its right. Also, assume that the size of the mask is much smaller than the image, so that image border effects are not a concern near the center of the horizontal intensity profile.

★**10.5** Suppose that we had used the edge model shown, instead of the ramp model of an edge shown in Fig. 10.6. Sketch the gradient and Laplacian of each profile.



10.6 Refer to Fig. 10.8 in answering the following questions.

(a) Assume that the Sobel masks are used to obtain G_x and G_y. Show that in this case the gradient computed by using Eqs. (10.1-4) and (10.1-12) give identical results for edges oriented in the horizontal and vertical directions.

(b) Show that this is true also for the Prewitt masks.

★**10.7** Show that the Sobel and Prewitt gradient masks of Figs. 10.8 and 10.9 give isotropic results only for horizontal and vertical edges, and for edges oriented at ±45°.

10.8 The results obtained by a single pass through an image of some 2-D masks can be achieved also by two passes using 1-D masks. For example, the same result of using a 3 × 3 smoothing mask with coefficients 1/9 can be obtained by a pass of

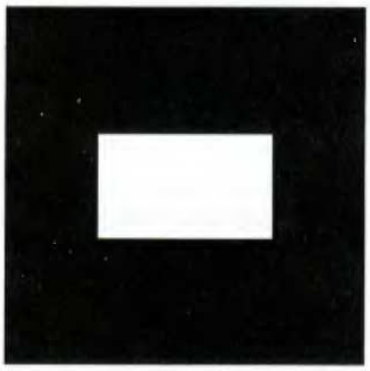
the mask [1 1 1] through an image. The result of this pass is then followed by a pass of the mask

$$\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}.$$

The final result is then scaled by 1/9. Show that the Sobel masks (Fig. 10.8) can be implemented by one pass of a *differencing* mask of the form [−1 0 1] (or its vertical counterpart) followed by a *smoothing* mask of the form [1 2 1] (or its vertical counterpart).

★**10.9** The so-called *compass gradient operators* of size 3×3 are designed to measure gradients of edges oriented in eight directions: E, NE, N, NW, W, SW, S, and SE. Give the form of these eight operators using coefficients valued 0, 1, or −1. Specify the gradient direction of each mask, keeping in mind that the gradient direction is orthogonal to the edge direction.

10.10 The center rectangle in the binary image shown is of size $m \times n$ pixels.



(a) Sketch the gradient of this image using the approximation given in Eq. (10.1-12). Assume that G_x and G_y are obtained by using the Sobel operators. Show all relevant different pixel values in the gradient image.

(b) Sketch the histogram of edge directions computed from Eq. (10.1-5). Be precise in labeling the height of each peak of the histogram.

(c) Sketch the Laplacian of the image for the approximation given in Eq. (10.1-14). Show all relevant different pixel values in the Laplacian image.

10.11 With reference to Eq. (10.1-17),

★ **(a)** Show that the average value of the Laplacian operator $\nabla^2 h$ is zero.

(b) Prove that the average value of any image convolved with this operator also is zero.

(c) Would (b) be true in general for the approximations to the Laplacian given in Eqs. (10.1-14) and (10.1-15)? Explain.

10.12 Refer to Fig. 10.15.

(a) Explain why the edges in Fig. 10.15(g) form closed contours.

(b) Does the zero-crossings method for edge finding always result in edges that are closed contours? Give a reason for your answer.

★10.13 Refer to the discussion in Section 10.2.2.

(a) Explain why the Hough mapping of point 1 in Fig. 10.20(b) is a straight line.

(b) Is this the only point that would produce this result?

(c) Explain the reflective adjacency relationship illustrated in Fig. 10.20(d).

10.14 Refer to the discussion in Section 10.2.2.

(a) Develop a general procedure for obtaining the normal representation of a line from its slope-intercept equation $y = ax + b$.

(b) Find the normal representation of the line $y = -2x + 1$.

10.15 An important area of application for image segmentation techniques is in processing images resulting from so-called bubble chamber events. These images arise from experiments in high-energy physics in which a beam of particles of known properties is directed onto a target of known nuclei. A typical event consists of incoming tracks, any one of which, in the event of a collision, branches out into secondary tracks of particles emanating from the point of collision. Propose a segmentation approach for detecting all tracks that contain at least 100 pixels and are angled at any of the following six directions off the horizontal: $\pm 25°$, $\pm 50°$, and $\pm 75°$. The allowable estimation error in any of these six directions is $\pm 5°$. For a track to be valid it must be at least 100 pixels long and not have more than three gaps, any of which cannot exceed 10 pixels. You may assume that the images have been preprocessed so that they are binary and that all tracks are 1 pixel wide, except at the point of collision from which they emanate. Your procedure should be able to differentiate between tracks that have the same direction but different origins. (*Hint*: Base your solution on the Hough transform.)

★10.16 Refer to Figs. 10.22 and 10.24.

(a) Superimpose on Fig. 10.22 all the possible edges given by the graph in Fig. 10.24.

(b) Compute the cost of the minimum-cost path.

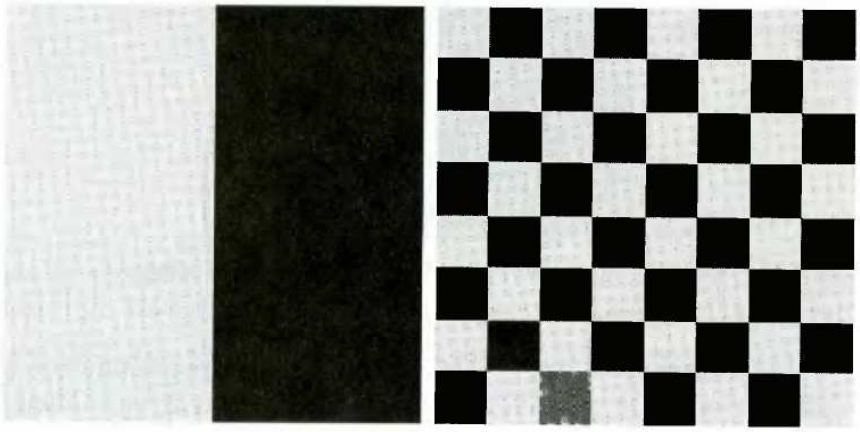
10.17 Find the edge corresponding to the minimum-cost path in the subimage shown. The numbers in brackets are gray levels and the outer numbers are spatial coordinates. Assume that the edge starts in the first column and ends in the last column.

	1	2	3
1	• [2]	• [1]	• [0]
2	• [1]	• [1]	• [7]
3	• [6]	• [8]	• [2]

★ **10.18** The images shown are quite different, but their histograms are identical. Suppose that each image is blurred with a 3×3 smoothing mask.

(a) Would the histograms still be identical after blurring?

(b) If your answer is no, sketch the two histograms.



10.19 Consider a noiseless image of size $N \times N$ whose first $N/2$ columns have gray level L_A and its remaining columns have gray level L_B, where $L_B > L_A$. The histogram of this image has only two peaks, of identical height, one located at L_A and the other at L_B. Segmenting this image into two halves based on its gray-level content is a trivial task that can be accomplished by a single global threshold located between L_A and L_B. Suppose, however, that you multiply the image by a gray-scale wedge that varies from 0 on the left to K on the right, with $K > L_B$. What would the histogram of this new image look like? Label the various parts of the histogram clearly.

10.20 Refer to the threshold finding algorithm introduced in Section 10.3.3. Assume that in a given problem the histogram is bimodal and, furthermore, that the shape of the modes can be approximated by Gaussian curves of the form $A_1 e^{-(z-m_1)^2/2\sigma_1^2}$ and $A_2 e^{-(z-m_2)^2/2\sigma_2^2}$. Assume that $m_1 < m_2$ and give conditions (in terms of the parameters of these curves) for the following to be true when the algorithm converges:

★ **(a)** The threshold is equal to $(m_1 + m_2)/2$.

★ **(b)** The threshold is to the left of m_1.

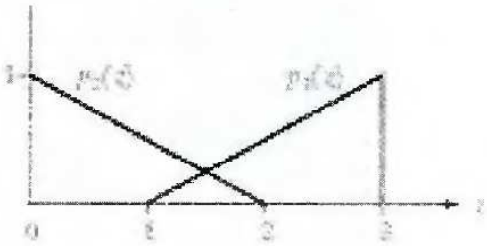
(c) The threshold is in the interval: $(m_1 + m_2)/2 < T < m_2$.

If it is impossible for any one of these conditions to exist, so state, and give a reason.

10.21 The illumination, $i(x, y)$, of a scene is known to vary as a function of spatial coordinates according to the equation $i(x, y) = A\left(1 + e^{-B[(x-N/2)^2+(y-N/2)^2]}\right)$, measured in some appropriately normalized units. The digitized scene, denoted $f(x, y)$, is of size $N \times N$, with $N = 1000$. The constants have values $A = 0.5$ and $B = 10^{-4}$. It is known that any subimage of $f(x, y)$ having a size greater than 10×10 pixels can be properly segmented, as long as the *illumination* spanning the area of the subimage did not vary by more than 0.1 units between any two points in the subimage when $f(x, y)$ was acquired. It is known also that the thresholds that work for the segmentation of the subimages under these conditions can be obtained using the algorithm introduced in Section 10.3.3. Propose an adaptive technique capable of thresholding this image. Not all subimages need to be

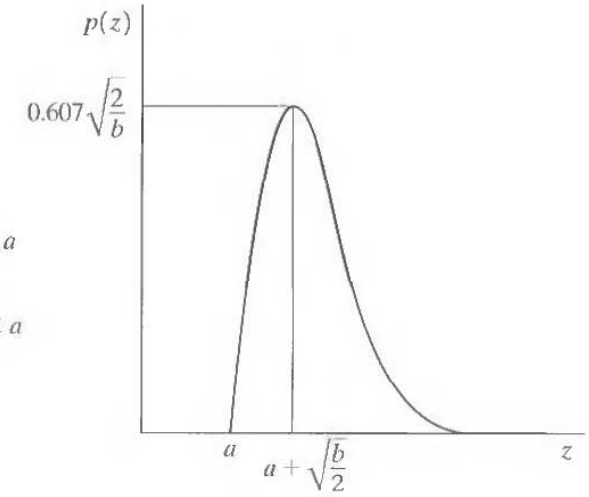
of the same size, but you are required to use the largest possible subimage at any location in the image.

★10.22 Suppose that an image has the gray-level probability density functions shown. Here, $p_1(z)$ corresponds to objects and $p_2(z)$ corresponds to the background. Assume that $P_1 = P_2$ and find the optimal threshold between object and background pixels.



10.23 The functional form of the Rayleigh probability density function and a typical plot are shown in the following figure. This density is well suited for approximating skewed histograms, such as the ones shown in Fig. 10.29. Use the Rayleigh density to set up a two-category bimodal problem, as in Problem 10.22. Find the optimal threshold in terms of the a priori probabilities P_1 and P_2, and the parameters of the Rayleigh density. The long "tails" of the two densities should be in opposite directions, as in Fig. 10.29.

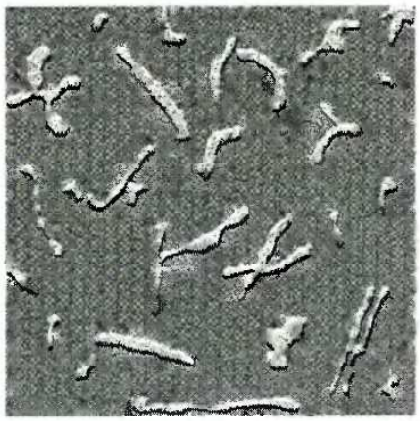
$$p(z) = \begin{cases} \frac{2}{b}(z-a)e^{-(z-a)^2/b} & z \geq a \\ 0 & z < a \end{cases}$$



★10.24 Start with Eq. (10.3-10) and derive Eqs. (10.3-12) and (10.3-13).

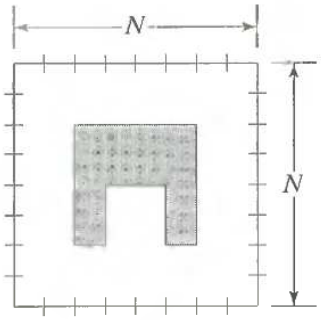
10.25 Derive Eq. (10.3-14) starting with Eqs. (10.3-12) and (10.3-13).

★10.26 The mean and standard deviation of the background pixels in the image shown are 110 and 15, respectively. The object pixels have mean and standard deviation values of 200 and 40, respectively. Propose a thresholding solution for segmenting the objects out of the image. State clearly any assumptions that you make in solving this problem.



10.27 Propose an approach based on region growing for segmenting the image shown in Problem 10.26. State clearly any assumptions that you make in solving this problem.

10.28 Segment the image shown by using the split and merge procedure discussed in Section 10.4.3. Let $P(R_i)$ = TRUE if all pixels in R_i have the same gray level. Show the quadtree corresponding to your segmentation.



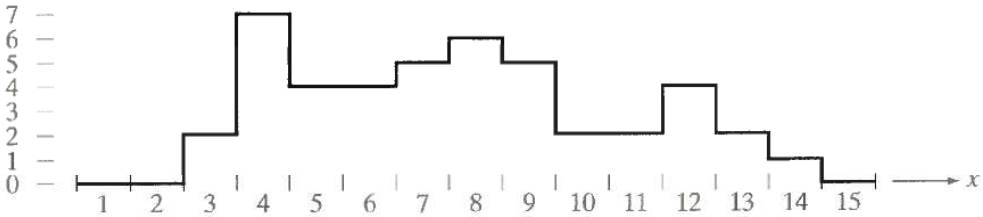
10.29 Refer to the discussion in Section 10.5.3.

★ **(a)** Show that the elements in sets $C_n(M_i)$ and $T[n]$ are never replaced during execution of the watershed algorithm.

(b) Show that the number of elements in sets $C_n(M_i)$ and $T[n]$ either increases or remains the same as n increases.

10.30 The boundaries illustrated in Section 10.5, obtained using the watershed segmentation algorithm, form closed loops. Advance an argument that establishes whether or not closed boundaries always result from application of this algorithm.

★**10.31** Give a step-by-step implementation of the dam building procedure for the one-dimensional gray-level cross section shown. Show a drawing of the cross section at each step, showing "water" levels and dams constructed.



10.32 What would the negative ADI image in Fig. 10.49(c) look like if we tested against T, instead of testing against $-T$, in Eq. (10.6-4)?

10.33 Determine if the following statements are true of false. Explain the reason for your answer.

(a) The nonzero entries of the absolute ADI continue to grow in dimension as long as the object is moving.

(b) The nonzero entries in the positive ADI always occupy the same area, regardless of the motion undergone by the object.

(c) The nonzero entries in the negative ADI continue to grow in dimension as long as the object is moving.

10.34 Suppose that in Example 10.21 motion along the x axis is set to zero. The object now moves only along the y axis at 1 pixel per frame for 32 frames and then (instantaneously) reverses direction and moves in exactly the opposite direction for another 32 frames. What would Figs. 10.53 and 10.54 look like under these conditions?

10.35 The speed of a bullet in flight is to be estimated by using high-speed imaging techniques. The method of choice involves the use of a TV camera and flash that exposes the scene for K s. The bullet is 2.5 cm long, 1 cm wide, and its range of speed is 750 ± 250 m/s. The camera optics produce an image in which the bullet occupies 10% of the horizontal resolution of a 256×256 digital image.

(a) Determine the maximum value of K that will guarantee that the blur from motion does not exceed 1 pixel.

(b) Determine the minimum number of frames per second that would have to be taken in order to guarantee that at least two complete images of the bullet are obtained during its path through the field of view of the camera.

(c) Propose a segmentation procedure for automatically extracting the bullet from a sequence of frames.

(d) Propose a method for automatically determining the speed of the bullet.

11 Representation and Description

Well, but reflect; have we not several times acknowledged that names rightly given are the likenesses and images of the things which they name?

Socrates

Preview

After an image has been segmented into regions by methods such as those discussed in Chapter 10, the resulting aggregate of segmented pixels usually is represented and described in a form suitable for further computer processing. Basically, representing a region involves two choices: (1) We can represent the region in terms of its external characteristics (its boundary), or (2) we can represent it in terms of its internal characteristics (the pixels comprising the region). Choosing a representation scheme, however, is only part of the task of making the data useful to a computer. The next task is to *describe* the region based on the chosen representation. For example, a region may be *represented* by its boundary, and the boundary *described* by features such as its length, the orientation of the straight line joining its extreme points, and the number of concavities in the boundary.

An external representation is chosen when the primary focus is on shape characteristics. An internal representation is selected when the primary focus is on regional properties, such as color and texture. Sometimes it may be necessary to use both types of representation. In either case, the features selected as descriptors should be as insensitive as possible to variations in size, translation, and rotation. For the most part, the descriptors discussed in this chapter satisfy one or more of these properties.

11.1 Representation

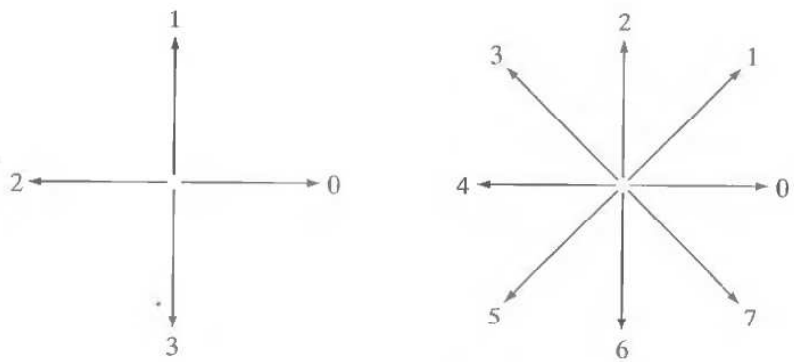
The segmentation techniques discussed in Chapter 10 yield raw data in the form of pixels along a boundary or pixels contained in a region. Although these data sometimes are used directly to obtain descriptors (as in determining the texture of a region), standard practice is to use schemes that compact the data into representations that are considerably more useful in the computation of descriptors. In this section we discuss various representation approaches.

11.1.1 Chain Codes

Chain codes are used to represent a boundary by a connected sequence of straight-line segments of specified length and direction. Typically, this representation is based on 4- or 8-connectivity of the segments. The direction of each segment is coded by using a numbering scheme such as the ones shown in Fig. 11.1.

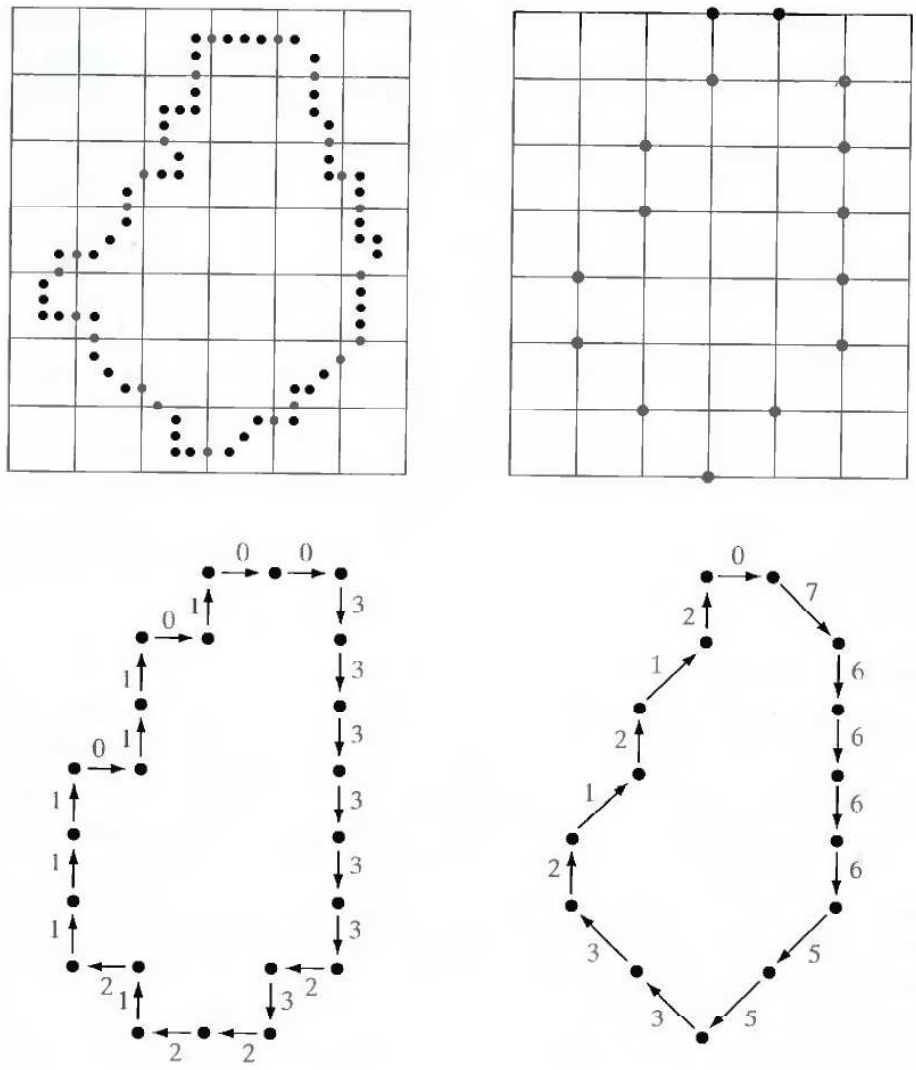
Digital images usually are acquired and processed in a grid format with equal spacing in the x- and y-directions, so a chain code could be generated by following a boundary in, say, a clockwise direction and assigning a direction to the segments connecting every pair of pixels. This method generally is unacceptable for two principal reasons: (1) The resulting chain of codes tends to be quite long, and (2) any small disturbances along the boundary due to noise or imperfect segmentation cause changes in the code that may not be related to the shape of the boundary.

An approach frequently used to circumvent the problems just discussed is to resample the boundary by selecting a larger grid spacing, as illustrated in Fig. 11.2(a). Then, as the boundary is traversed, a boundary point is assigned to each node of the large grid, depending on the proximity of the original boundary to that node, as shown in Fig. 11.2(b). The resampled boundary obtained in this way then can be represented by a 4- or 8-code, as shown in Figs. 11.2(c) and (d), respectively. The starting point in Fig. 11.2(c) is (arbitrarily) at the top, left dot, and the boundary is the shortest allowable 4- or 8-path in the grid of Fig. 11.2(b). The boundary representation in Fig. 11.2(c) is the chain code 0033 ... 01, and in Fig. 11.2(d) it is the code 0766 ... 12. As might be expected, the accuracy of the resulting code representation depends on the spacing of the sampling grid.



a b

FIGURE 11.1 Direction numbers for (a) 4-directional chain code, and (b) 8-directional chain code.



a b
c d

FIGURE 11.2 (a) Digital boundary with resampling grid superimposed. (b) Result of resampling. (c) 4-directional chain code. (d) 8-directional chain code.

The chain code of a boundary depends on the starting point. However, the code can be normalized with respect to the starting point by a straightforward procedure: We simply treat the chain code as a circular sequence of direction numbers and redefine the starting point so that the resulting sequence of numbers forms an integer of minimum magnitude. We can normalize also for rotation by using the *first difference* of the chain code instead of the code itself. This difference is obtained by counting the number of direction changes (in a counterclockwise direction) that separate two adjacent elements of the code. For instance, the first difference of the 4-direction chain code 10103322 is 3133030. If we elect to treat the code as a circular sequence, then the first element of the difference is computed by using the transition between the last and first components of the chain. Here, the result is 33133030. Size normalization can be achieved by altering the size of the resampling grid.

These normalizations are exact only if the boundaries themselves are invariant to rotation and scale change, which, in practice, is seldom the case. For instance, the same object digitized in two different orientations will in general have

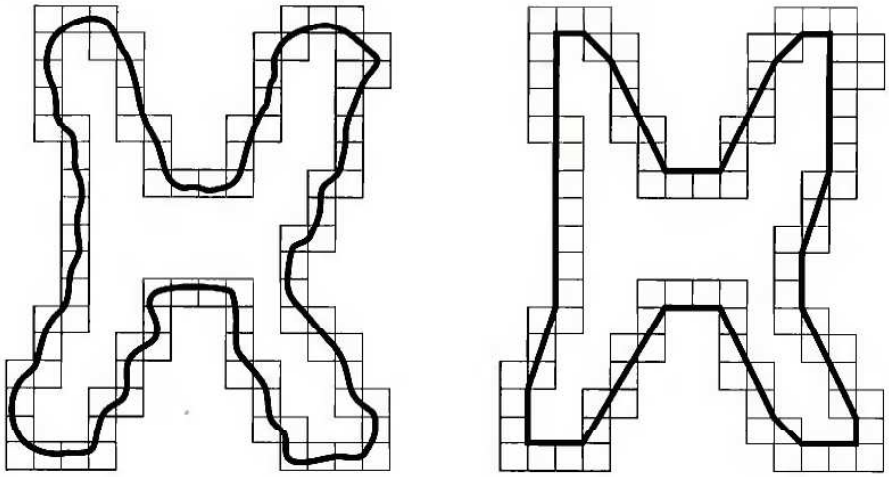
different boundary shapes, with the degree of dissimilarity being proportional to image resolution. This effect can be reduced by selecting chain elements that are large in proportion to the distance between pixels in the digitized image and/or by orienting the resampling grid along the principal axes of the object to be coded, as discussed in Section 11.2.2, or along its eigen axes, as discussed in Section 11.4.

11.1.2 Polygonal Approximations

A digital boundary can be approximated with arbitrary accuracy by a polygon. For a closed curve, the approximation is exact when the number of segments in the polygon is equal to the number of points in the boundary so that each pair of adjacent points defines a segment in the polygon. In practice, the goal of polygonal approximation is to capture the "essence" of the boundary shape with the fewest possible polygonal segments. This problem in general is not trivial and can quickly turn into a time-consuming iterative search. However, several polygonal approximation techniques of modest complexity and processing requirements are well suited for image processing applications.

Minimum perimeter polygons

We begin the discussion of polygonal approximations with a method for finding *minimum perimeter polygons*. The procedure is best explained by an example. Suppose that we enclose a boundary by a set of concatenated cells, as shown in Fig. 11.3(a). It helps to visualize this enclosure as two walls corresponding to the outside and inside boundaries of the strip of cells, and think of the object boundary as a rubber band contained within the walls. If the rubber band is allowed to shrink, it takes the shape shown in Fig. 11.3(b), producing a polygon of minimum perimeter that fits the geometry established by the cell strip. If each cell encompasses only one point on the boundary, the error in each cell between the original boundary and the rubber-band approximation at most would be $\sqrt{2}d$, where d is the minimum possible distance between different pixels (i.e., the distance between lines in the sampling grid used to produce the digital image). This error can be reduced by half by forcing each cell to be centered on its corresponding pixel.



a b

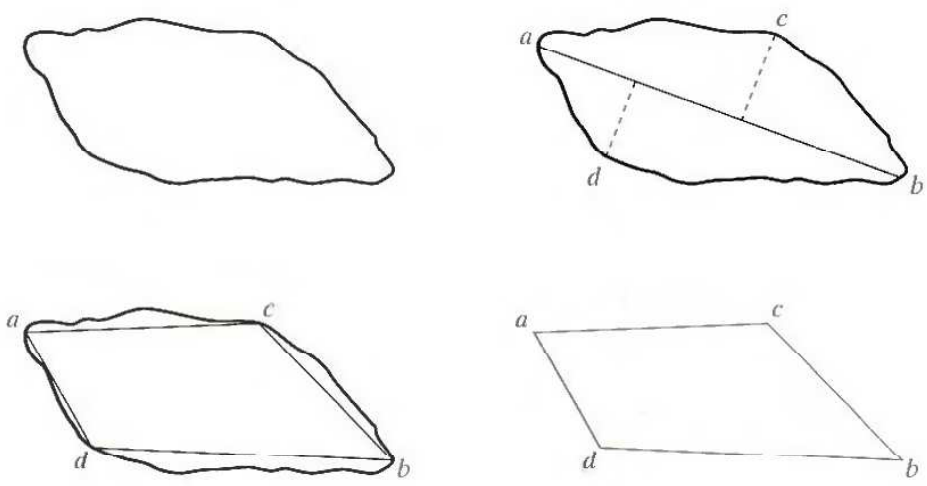
FIGURE 11.3 (a) Object boundary enclosed by cells. (b) Minimum perimeter polygon.

Merging techniques

Merging techniques based on average error or other criteria have been applied to the problem of polygonal approximation. One approach is to merge points along a boundary until the least square error line fit of the points merged so far exceeds a preset threshold. When this condition occurs, the parameters of the line are stored, the error is set to 0, and the procedure is repeated, merging new points along the boundary until the error again exceeds the threshold. At the end of the procedure the intersections of adjacent line segments form the vertices of the polygon. One of the principal difficulties with this method is that vertices in the resulting approximation do not always correspond to inflections (such as corners) in the original boundary, because a new line is not started until the error threshold is exceeded. If, for instance, a long straight line were being tracked and it turned a corner, a number (depending on the threshold) of points past the corner would be absorbed before the threshold was exceeded. However, splitting (discussed next) along with merging may be used to alleviate this difficulty.

Splitting techniques

One approach to boundary segment *splitting* is to subdivide a segment successively into two parts until a specified criterion is satisfied. For instance, a requirement might be that the maximum perpendicular distance from a boundary segment to the line joining its two end points not exceed a preset threshold. If it does, the farthest point from the line becomes a vertex, thus subdividing the initial segment into two subsegments. This approach has the advantage of seeking prominent inflection points. For a closed boundary, the best starting points usually are the two farthest points in the boundary. For example, Fig. 11.4(a) shows an object boundary, and Fig. 11.4(b) shows a subdivision of this boundary (solid line) about its farthest points. The point marked c is the farthest point (in terms of perpendicular distance) from the top boundary segment to line ab. Similarly, point d is the farthest point in the bottom segment. Figure 11.4(c) shows the result of using the splitting procedure with a threshold equal to 0.25 times the length of line ab. As no point in the new



a b
c d

FIGURE 11.4 (a) Original boundary. (b) Boundary divided into segments based on extreme points. (c) Joining of vertices. (d) Resulting polygon.

boundary segments has a perpendicular distance (to its corresponding straight-line segment) that exceeds this threshold, the procedure terminates with the polygon shown in Fig. 11.4(d).

11.1.3 Signatures

A signature is a 1-D functional representation of a boundary and may be generated in various ways. One of the simplest is to plot the distance from the centroid to the boundary as a function of angle, as illustrated in Fig. 11.5. Regardless of how a signature is generated, however, the basic idea is to reduce the boundary representation to a 1-D function, which presumably is easier to describe than the original 2-D boundary.

Signatures generated by the approach just described are invariant to translation, but they do depend on rotation and scaling. Normalization with respect to rotation can be achieved by finding a way to select the same starting point to generate the signature, regardless of the shape's orientation. One way to do so is to select the starting point as the point farthest from the centroid, if this point happens to be unique and independent of rotational aberrations for each shape of interest. Another way is to select the point on the eigen axis (see Section 11.4) that is farthest from the centroid. This method requires more computation but is more rugged because the direction of the eigen axis is determined by using all contour points. Yet another way is to obtain the chain code of the boundary and then use the approach discussed in Section 11.1.1, assuming that the coding is coarse enough so that rotation does not affect its circularity.

Based on the assumptions of uniformity in scaling with respect to both axes and that sampling is taken at equal intervals of θ, changes in size of a shape result in changes in the amplitude values of the corresponding signature. One way to normalize for this result is to scale all functions so that they always span the

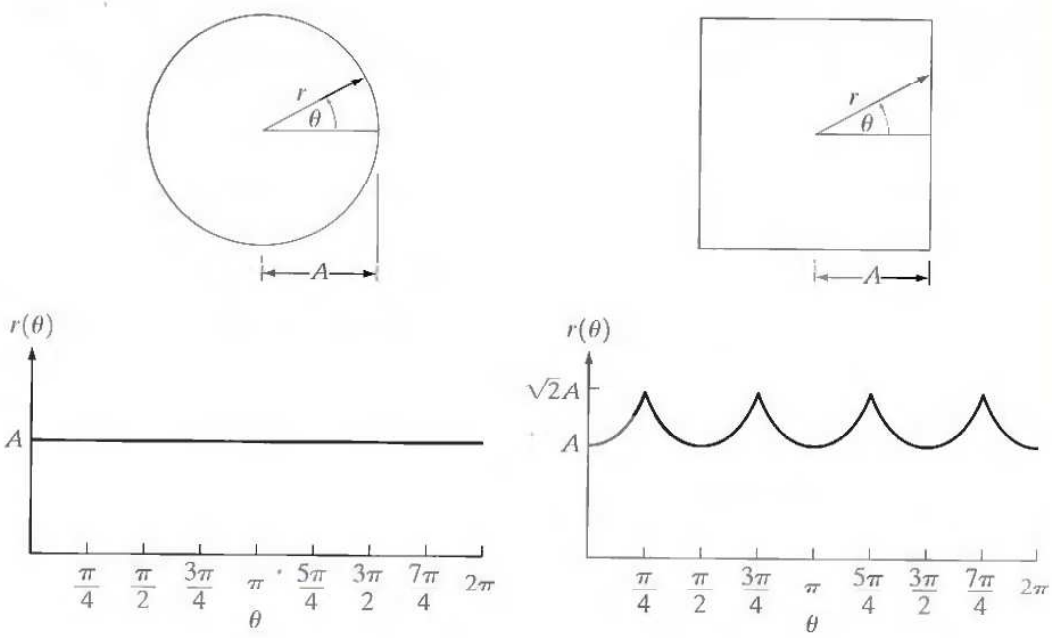


FIGURE 11.5 Distance-versus-angle signatures. In (a) $r(\theta)$ is constant. In (b), the signature consists of repetitions of the pattern $r(\theta) = A \sec\theta$ for $0 \le \theta \le \pi/4$ and $r(\theta) = A \csc\theta$ for $\pi/4 < \theta \le \pi/2$.

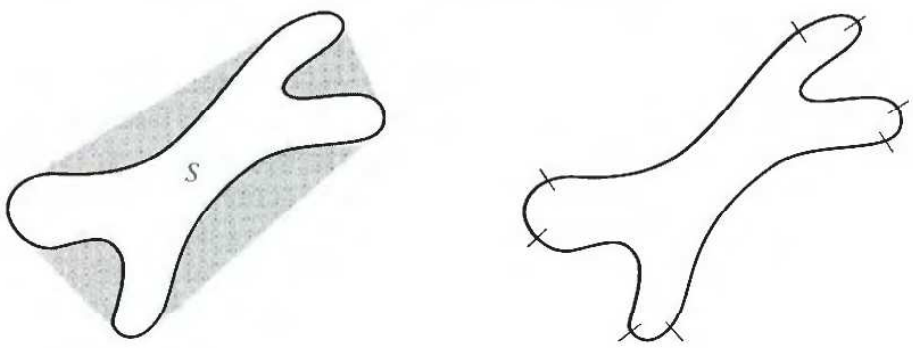
same range of values, say, [0, 1]. The main advantage of this method is simplicity, but it has the potentially serious disadvantage that scaling of the entire function depends on only two values: the minimum and maximum. If the shapes are noisy, this dependence can be a source of error from object to object. A more rugged (but also more computationally intensive) approach is to divide each sample by the variance of the signature, assuming that the variance is not zero—as in the case of Fig. 11.5(a)—or so small that it creates computational difficulties. Use of the variance yields a variable scaling factor that is inversely proportional to changes in size and works much as automatic gain control does. Whatever the method used, keep in mind that the basic idea is to remove dependency on size while preserving the fundamental shape of the waveforms.

Of course, distance versus angle is not the only way to generate a signature. For example, another way is to traverse the boundary and, corresponding to each point on the boundary, plot the angle between a line tangent to the boundary at that point and a reference line. The resulting signature, although quite different from the $r(\theta)$ curve, would carry information about basic shape characteristics. For instance, horizontal segments in the curve would correspond to straight lines along the boundary, because the tangent angle would be constant there. A variation of this approach is to use the so-called *slope density function* as a signature. This function is simply a histogram of tangent-angle values. As a histogram is a measure of concentration of values, the slope density function responds strongly to sections of the boundary with constant tangent angles (straight or nearly straight segments) and has deep valleys in sections producing rapidly varying angles (corners or other sharp inflections).

11.1.4 Boundary Segments

Decomposing a boundary into segments often is useful. Decomposition reduces the boundary's complexity and thus simplifies the description process. This approach is particularly attractive when the boundary contains one or more significant concavities that carry shape information. In this case use of the convex hull of the region enclosed by the boundary is a powerful tool for robust decomposition of the boundary.

As defined in Section 9.5.4, the *convex hull* H of an arbitrary set S is the smallest convex set containing S. The set difference $H - S$ is called the *convex deficiency* D of the set S. To see how these concepts might be used to partition a boundary into meaningful segments, consider Fig. 11.6(a), which shows an



a b

FIGURE 11.6 (a) A region, S, and its convex deficiency (shaded). (b) Partitioned boundary.

object (set S) and its convex deficiency (shaded regions). The region boundary can be partitioned by following the contour of S and marking the points at which a transition is made into or out of a component of the convex deficiency. Figure 11.6(b) shows the result in this case. Note that in principle, this scheme is independent of region size and orientation.

In practice, digital boundaries tend to be irregular because of digitization, noise, and variations in segmentation. These effects usually result in convex deficiencies that have small, meaningless components scattered randomly throughout the boundary. Rather than attempt to sort out these irregularities by postprocessing, a common approach is to smooth a boundary prior to partitioning. There are a number of ways to do so. One way is to traverse the boundary and replace the coordinates of each pixel by the average coordinates of k of its neighbors along the boundary. This approach works for small irregularities, but it is time-consuming and difficult to control. Large values of k can result in excessive smoothing, whereas small values of k might not be sufficient in some segments of the boundary. A more rugged technique is to use a polygonal approximation, as discussed in Section 11.1.2, prior to finding the convex deficiency of a region. Most digital boundaries of interest are simple polygons (polygons without self-intersection). Graham and Yao [1983] give an algorithm for finding the convex hull of such polygons.

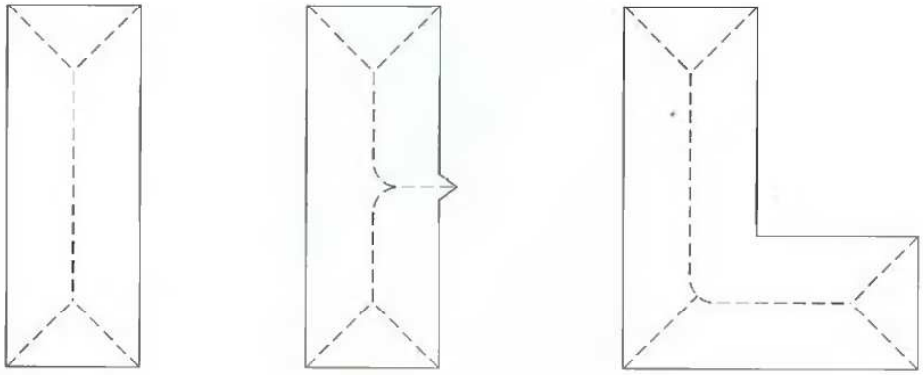
The concepts of a convex hull and its deficiency are equally useful for describing an entire region, as well as just its boundary. For example, description of a region might be based on its area and the area of its convex deficiency, the number of components in the convex deficiency, the relative location of these components, and so on. Recall that a morphological algorithm for finding the convex hull was developed in Section 9.5.4. References cited at the end of this chapter contain other formulations.

11.1.5 Skeletons

An important approach to representing the structural shape of a plane region is to reduce it to a graph. This reduction may be accomplished by obtaining the *skeleton* of the region via a thinning (also called *skeletonizing*) algorithm. Thinning procedures play a central role in a broad range of problems in image processing, ranging from automated inspection of printed circuit boards to counting of asbestos fibers in air filters. We already discussed in Section 9.5.7 the basics of skeletonizing using morphology. However, as noted in that section, the procedure discussed there made no provisions for keeping the skeleton connected. The algorithm developed here corrects that problem.

The skeleton of a region may be defined via the medial axis transformation (MAT) proposed by Blum [1967]. The MAT of a region R with border B is as follows. For each point p in R, we find its closest neighbor in B. If p has more than one such neighbor, it is said to belong to the *medial axis* (skeleton) of R. The concept of "closest" (and the resulting MAT) depend on the definition of a distance (see Section 2.5.3). Figure 11.7 shows some examples using the Euclidean distance. The same results would be obtained with the maximum disk of Section 9.5.7.

The MAT of a region has an intuitive definition based on the so-called "prairie fire concept." Consider an image region as a prairie of uniform, dry



a b c

FIGURE 11.7 Medial axes (dashed) of three simple regions.

grass, and suppose that a fire is lit along its border. All fire fronts will advance into the region at the same speed. The MAT of the region is the set of points reached by more than one fire front at the same time.

Although the MAT of a region yields an intuitively pleasing skeleton, direct implementation of this definition typically is expensive computationally. Implementation potentially involves calculating the distance from every interior point to every point on the boundary of a region. Numerous algorithms have been proposed for improving computational efficiency while at the same time attempting to produce a medial axis representation of a region. Typically, these are thinning algorithms that iteratively delete edge points of a region subject to the constraints that deletion of these points (1) does not remove end points, (2) does not break connectivity, and (3) does not cause excessive erosion of the region.

In this section we present an algorithm for thinning binary regions. Region points are assumed to have value 1 and background points to have value 0. The method consists of successive passes of two basic steps applied to the contour points of the given region, where, based on the definition given in Section 2.5.2, a *contour point* is any pixel with value 1 and having at least one 8-neighbor valued 0. With reference to the 8-neighborhood notation shown in Fig. 11.8, step 1 flags a contour point p_1 for deletion if the following conditions are satisfied:

(a) $2 \leq N(p_1) \leq 6$
(b) $T(p_1) = 1$
(c) $p_2 \cdot p_4 \cdot p_6 = 0$
(d) $p_4 \cdot p_6 \cdot p_8 = 0$ (11.1-1)

where $N(p_1)$ is the number of nonzero neighbors of p_1; that is,

$$N(p_1) = p_2 + p_3 + \cdots + p_8 + p_9 \tag{11.1-2}$$

p_9	p_2	p_3
p_8	p_1	p_4
p_7	p_6	p_5

FIGURE 11.8 Neighborhood arrangement used by the thinning algorithm.

FIGURE 11.9 Illustration of conditions (a) and (b) in Eq. (11.1-1). In this case $N(p_1) = 4$ and $T(p_1) = 3$.

0	0	1
1	p_1	0
1	0	1

and $T(p_1)$ is the number of 0-1 transitions in the ordered sequence $p_2, p_3, \ldots, p_8, p_9, p_2$. For example, $N(p_1) = 4$ and $T(p_1) = 3$ in Fig. 11.9.

In step 2, conditions (a) and (b) remain the same, but conditions (c) and (d) are changed to

$$\begin{aligned} &\textbf{(c')}\ p_2 \cdot p_4 \cdot p_8 = 0 \\ &\textbf{(d')}\ p_2 \cdot p_6 \cdot p_8 = 0 \end{aligned} \tag{11.1-3}$$

Step 1 is applied to every border pixel in the binary region under consideration. If one or more of conditions (a)–(d) are violated, the value of the point in question is not changed. If all conditions are satisfied the point is flagged for deletion. However, the point is not deleted until all border points have been processed. This delay prevents changing the structure of the data during execution of the algorithm. After step 1 has been applied to all border points, those that were flagged are deleted (changed to 0). Then step 2 is applied to the resulting data in exactly the same manner as step 1.

Thus one iteration of the thinning algorithm consists of (1) applying step 1 to flag border points for deletion; (2) deleting the flagged points; (3) applying step 2 to flag the remaining border points for deletion; and (4) deleting the flagged points. This basic procedure is applied iteratively until no further points are deleted, at which time the algorithm terminates, yielding the skeleton of the region.

Condition (a) is violated when contour point p_1 only has one or seven 8-neighbors valued 1. Having only one such neighbor implies that p_1 is the end point of a skeleton stroke and obviously should not be deleted. Deleting p_1 if it had seven such neighbors would cause erosion into the region. Condition (b) is violated when it is applied to points on a stroke 1 pixel thick. Hence this condition prevents disconnection of segments of a skeleton during the thinning operation. Conditions (c) and (d) are satisfied simultaneously by the minimum set of values: ($p_4 = 0$ *or* $p_6 = 0$) or ($p_2 = 0$ *and* $p_8 = 0$). Thus with reference to the neighborhood arrangement in Fig. 11.8, a point that satisfies these conditions, as well as conditions (a) and (b), is an east or south boundary point or a northwest corner point in the boundary. In either case, p_1 is not part of the skeleton and should be removed. Similarly, conditions (c′) and (d′) are satisfied simultaneously by the following minimum set of values: ($p_2 = 0$ *or* $p_8 = 0$) or ($p_4 = 0$ *and* $p_6 = 0$). These correspond to north or west boundary points, or a southeast corner point. Note that northeast corner points have $p_2 = 0$ and $p_4 = 0$, and thus satisfy conditions (c) and (d), as well as (c′) and (d′). The same is true for southwest corner points, which have $p_6 = 0$ and $p_8 = 0$.

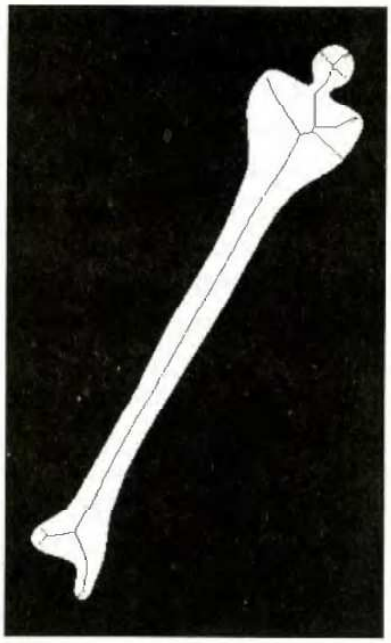


FIGURE 11.10 Human leg bone and skeleton of the region shown superimposed.

EXAMPLE 11.1: The skeleton of a region.

■ Figure 11.10 shows a segmented image of a human leg bone and, superimposed, the skeleton of the region computed using the algorithm just discussed. For the most part, the skeleton looks intuitively correct. There is a double branch on the right side of the "shoulder" of the bone that at first glance one would expect to be a single branch, as on the corresponding left side. Note, however, that the right shoulder is somewhat broader (in the long direction) than the left shoulder. That is what caused the branch to be created by the algorithm. This type of unpredictable behavior is not unusual in skeletonizing algorithms. ■

11.2 Boundary Descriptors

In this section we consider several approaches to describing the boundary of a region, and in Section 11.3 we focus on regional descriptors. Parts of Sections 11.4 and 11.5 are applicable to both boundaries and regions.

11.2.1 Some Simple Descriptors

The *length* of a boundary is one of its simplest descriptors. The number of pixels along a boundary gives a rough approximation of its length. For a chain-coded curve with unit spacing in both directions, the number of vertical and horizontal components plus $\sqrt{2}$ times the number of diagonal components gives its exact length.

The *diameter* of a boundary B is defined as

$$\text{Diam}(B) = \max_{i,j}\left[D(p_i, p_j)\right] \tag{11.2-1}$$

where D is a distance measure (see Section 2.5.3) and p_i and p_j are points on the boundary. The value of the diameter and the orientation of a line segment connecting the two extreme points that comprise the diameter (this line is called the

major axis of the boundary) are useful descriptors of a boundary. The *minor axis* of a boundary is defined as the line perpendicular to the major axis, and of such length that a box passing through the outer four points of intersection of the boundary with the two axes completely encloses the boundary.† The box just described is called the *basic rectangle*, and the ratio of the major to the minor axis is called the *eccentricity* of the boundary. This also is a useful descriptor.

Curvature is defined as the rate of change of slope. In general, obtaining reliable measures of curvature at a point in a digital boundary is difficult because these boundaries tend to be locally "ragged." However, using the difference between the slopes of adjacent boundary segments (which have been represented as straight lines) as a descriptor of curvature at the point of intersection of the segments sometimes proves useful. For example, the vertices of boundaries such as those shown in Figs. 11.3(b) and 11.4(d) lend themselves well to curvature descriptions. As the boundary is traversed in the clockwise direction, a vertex point p is said to be part of a *convex* segment if the change in slope at p is nonnegative; otherwise, p is said to belong to a segment that is *concave*. The description of curvature at a point can be refined further by using ranges in the change of slope. For instance, p could be part of a nearly straight segment if the change is less than 10° or a *corner* point if the change exceeds 90°. Note, however, that these descriptors must be used with care because their interpretation depends on the length of the individual segments relative to the overall length of the boundary.

11.2.2 Shape Numbers

As explained in Section 11.1.1, the first difference of a chain-coded boundary depends on the starting point. The *shape number* of such a boundary, based on the 4-directional code of Fig. 11.1(a), is defined as the first difference of smallest magnitude. The *order* n of a shape number is defined as the number of digits in its representation. Moreover, n is even for a closed boundary, and its value limits the number of possible different shapes. Figure 11.11 shows all the shapes of order 4, 6, and 8, along with their chain-code representations, first differences, and corresponding shape numbers. Note that the first difference is computed by treating the chain code as a circular sequence, as discussed in Section 11.1.1. Although the first difference of a chain code is independent of rotation, in general the coded boundary depends on the orientation of the grid. One way to normalize the grid orientation is by aligning the chain-code grid with the sides of the basic rectangle defined in the previous section.

In practice, for a desired shape order, we find the rectangle of order n whose eccentricity (defined in the previous section) best approximates that of the basic rectangle and use this new rectangle to establish the grid size. For example, if $n = 12$, all the rectangles of order 12 (that is, those whose perimeter length is 12) are 2×4, 3×3, and 1×5. If the eccentricity of the 2×4 rectangle best matches the eccentricity of the basic rectangle for a given boundary, we establish a 2×4 grid centered on the basic rectangle and use the procedure outlined

† Do not confuse this definition of major and minor axes with the eigen axes, which are defined in Section 11.4.

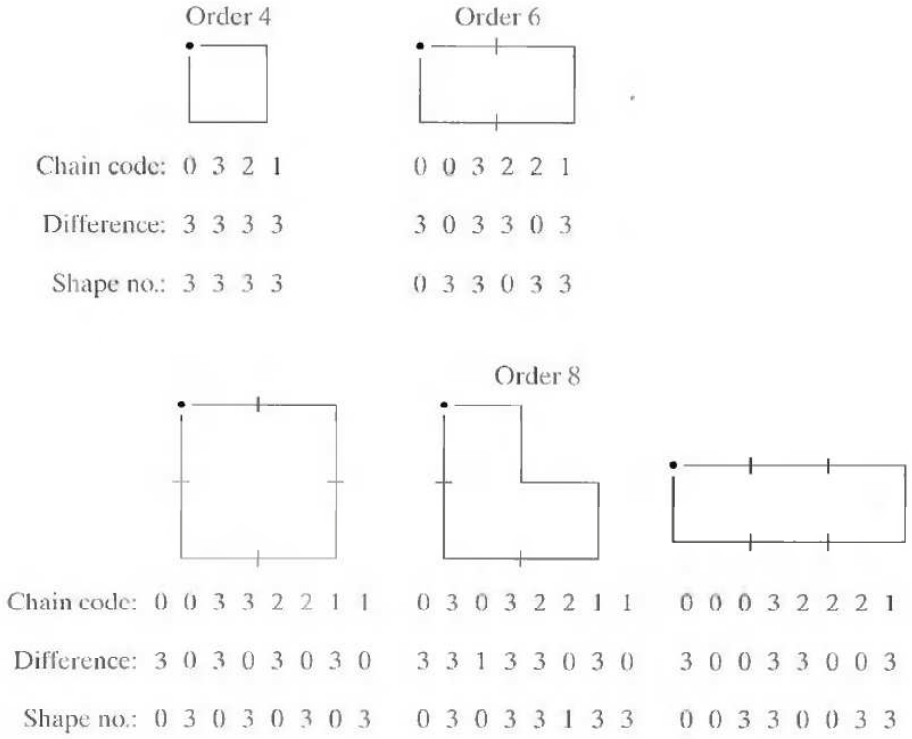


FIGURE 11.11 All shapes of order 4, 6, and 8. The directions are from Fig. 11.1(a), and the dot indicates the starting point.

in Section 11.1.1 to obtain the chain code. The shape number follows from the first difference of this code. Although the order of the resulting shape number usually equals n because of the way the grid spacing was selected, boundaries with depressions comparable to this spacing sometimes yield shape numbers of order greater than n. In this case, we specify a rectangle of order lower than n and repeat the procedure until the resulting shape number is of order n.

EXAMPLE 11.2: Computing shape numbers.

Suppose that $n = 18$ is specified for the boundary shown in Fig. 11.12(a). To obtain a shape number of this order requires following the steps just discussed. The first step is to find the basic rectangle, as shown in Fig. 11.12(b). The closest rectangle of order 18 is a 3×6 rectangle, requiring subdivision of the basic rectangle as shown in Fig. 11.12(c), where the chain-code directions are aligned with the resulting grid. The final step is to obtain the chain code and use its first difference to compute the shape number, as shown in Fig. 11.12(d).

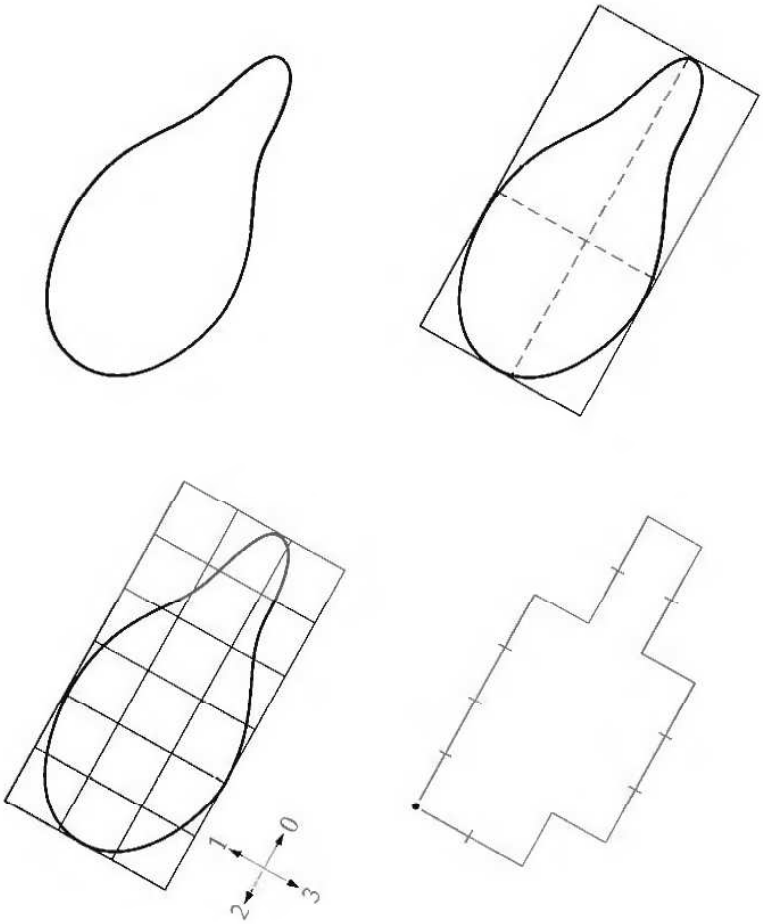
11.2.3 Fourier Descriptors

Figure 11.13 shows a K-point digital boundary in the xy-plane. Starting at an arbitrary point (x_0, y_0), coordinate pairs $(x_0, y_0), (x_1, y_1), (x_2, y_2), \ldots, (x_{K-1}, y_{K-1})$ are encountered in traversing the boundary, say, in the counterclockwise direction. These coordinates can be expressed in the form $x(k) = x_k$ and $y(k) = y_k$. With this notation, the boundary itself can be represented as the sequence of coordinates $s(k) = [x(k), y(k)]$, for $k = 0, 1, 2, \ldots, K - 1$. Moreover, each coordinate pair can be treated as a complex number so that

$$s(k) = x(k) + jy(k) \tag{11.2-2}$$

a b
c d

FIGURE 11.12 Steps in the generation of a shape number.



Chain code:	0	0	0	0	3	0	0	3	2	2	3	2	2	2	1	2	1	1
Difference:	3	0	0	0	3	1	0	3	3	0	1	3	0	0	3	1	3	0
Shape no.:	0	0	0	3	1	0	3	3	0	1	3	0	0	3	1	3	0	3

for $k = 0, 1, 2, \ldots, K - 1$. That is, the x-axis is treated as the real axis and the y-axis as the imaginary axis of a sequence of complex numbers. Although the interpretation of the sequence was recast, the nature of the boundary itself was not changed. Of course, this representation has one great advantage: It reduces a 2-D to a 1-D problem.

From Section 4.2.1, the discrete Fourier transform (DFT) of $s(k)$ is

$$a(u) = \frac{1}{K}\sum_{k=0}^{K-1} s(k)e^{-j2\pi uk/K} \tag{11.2-3}$$

for $u = 0, 1, 2, \ldots, K - 1$. The complex coefficients $a(u)$ are called the *Fourier descriptors* of the boundary. The inverse Fourier transform of these coefficients restores $s(k)$. That is,

$$s(k) = \sum_{u=0}^{K-1} a(u)e^{j2\pi uk/K} \tag{11.2-4}$$

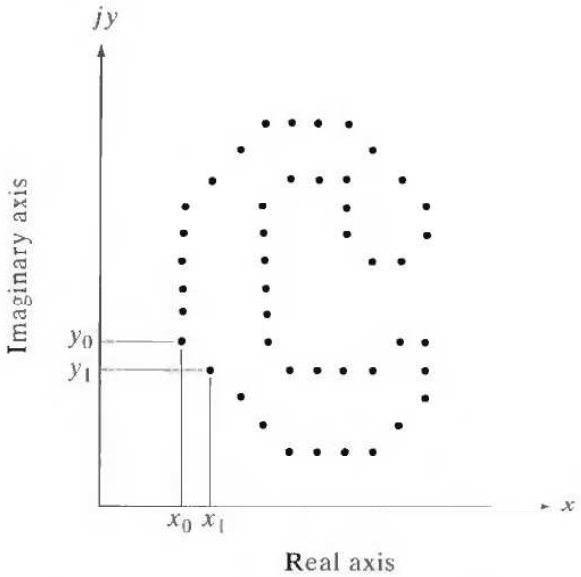


FIGURE 11.13 A digital boundary and its representation as a complex sequence. The points (x_0, y_0) and (x_1, y_1) shown are (arbitrarily) the first two points in the sequence.

for $k = 0, 1, 2, \ldots, K - 1$. Suppose, however, that instead of all the Fourier coefficients, only the first P coefficients are used. This is equivalent to setting $a(u) = 0$ for $u > P - 1$ in Eq. (11.2-4). The result is the following *approximation* to $s(k)$:

$$\hat{s}(k) = \sum_{u=0}^{P-1} a(u) e^{j2\pi uk/K} \tag{11.2-5}$$

for $k = 0, 1, 2, \ldots, K - 1$. Although only P terms are used to obtain each component of $\hat{s}(k)$, k still ranges from 0 to $K - 1$. That is, the *same* number of points exists in the approximate boundary, but not as many terms are used in the reconstruction of each point. Recall from discussions of the Fourier transform in Chapter 4 that high-frequency components account for fine detail, and low-frequency components determine global shape. Thus the smaller P becomes, the more detail that is lost on the boundary. The following example demonstrates this clearly.

EXAMPLE 11.3: Illustration of Fourier descriptors.

■ Figure 11.14 shows a square boundary consisting of $K = 64$ points and the results of using Eq. (11.2-5) to reconstruct this boundary for various values of P. Note that the value of P has to be about 8 before the reconstructed boundary looks more like a square than a circle. Next, note that little in the way of corner definition occurs until P is about 56, at which time the corner points begin to "break out" of the sequence. Finally, note that, when $P = 61$, the curves begin to straighten, which leads to an almost exact replica of the original one additional coefficient later. Thus, a few low-order coefficients are able to capture gross shape, but many more high-order terms are required to define accurately sharp features such as corners and straight lines. This result is not unexpected in view of the role played by low- and high-frequency components in defining the shape of a region. ■

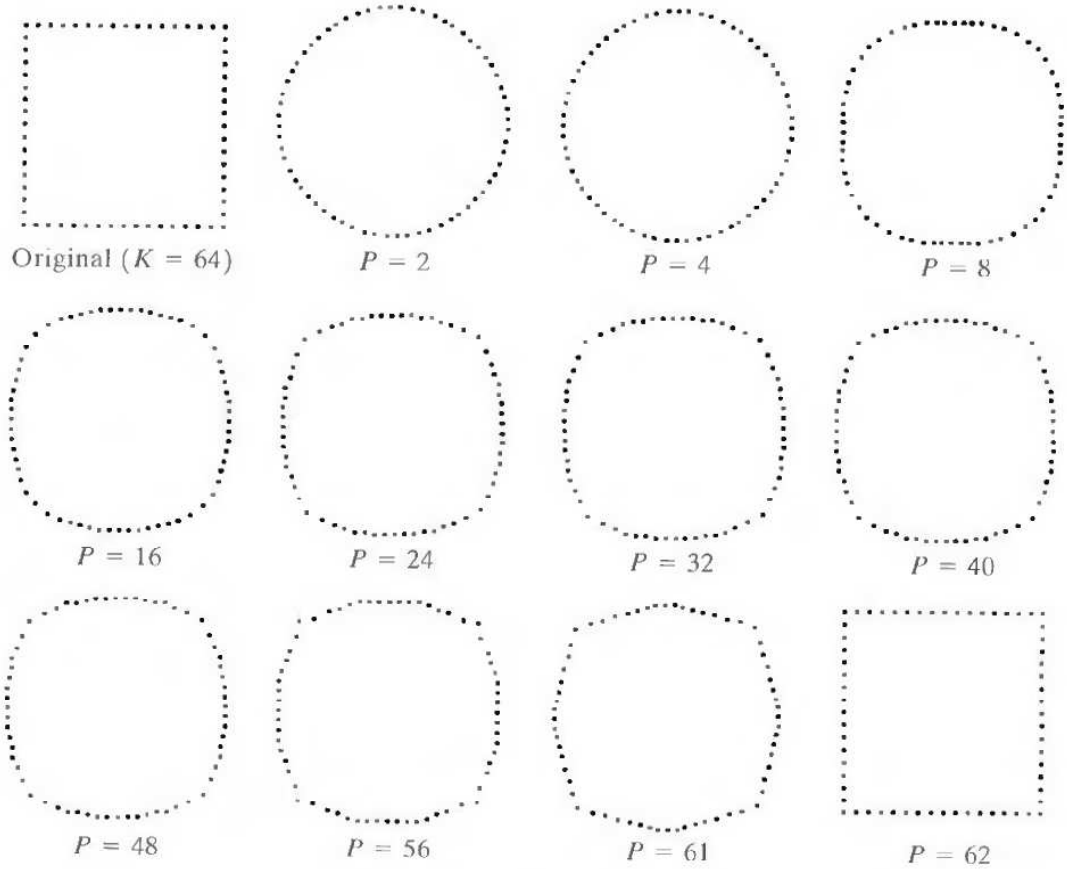


FIGURE 11.14 Examples of reconstruction from Fourier descriptors. P is the number of Fourier coefficients used in the reconstruction of the boundary.

As demonstrated in the preceding example, a few Fourier descriptors can be used to capture the gross essence of a boundary. This property is valuable, because these coefficients carry shape information. Thus they can be used as the basis for differentiating between distinct boundary shapes, as we discuss in some detail in Chapter 12.

We have stated several times that descriptors should be as insensitive as possible to translation, rotation, and scale changes. In cases where results depend on the order in which points are processed, an additional constraint is that descriptors should be insensitive to starting point. Fourier descriptors are not directly insensitive to these geometrical changes, but the changes in these parameters can be related to simple transformations on the descriptors. For example, consider rotation, and recall from elementary mathematical analysis that rotation of a point by an angle θ about the origin of the complex plane is accomplished by multiplying the point by $e^{j\theta}$. Doing so to every point of $s(k)$ rotates the entire sequence about the origin. The rotated sequence is $s(k)e^{j\theta}$, whose Fourier descriptors are

$$\begin{aligned} a_r(u) &= \frac{1}{K}\sum_{k=0}^{K-1} s(k)e^{j\theta}e^{-j2\pi uk/K} \\ &= a(u)e^{j\theta} \end{aligned} \tag{11.2-6}$$

for $u = 0, 1, 2, \ldots, K-1$. Thus rotation simply affects all coefficients equally by a multiplicative *constant* term $e^{j\theta}$.

TABLE 11.1 Some basic properties of Fourier descriptors.

Transformation	Boundary	Fourier Descriptor
Identity	$s(k)$	$a(u)$
Rotation	$s_r(k) = s(k)e^{j\theta}$	$a_r(u) = a(u)e^{j\theta}$
Translation	$s_t(k) = s(k) + \Delta_{xy}$	$a_t(u) = a(u) + \Delta_{xy}\delta(u)$
Scaling	$s_s(k) = \alpha s(k)$	$a_s(u) = \alpha a(u)$
Starting point	$s_p(k) = s(k - k_0)$	$a_p(u) = a(u)e^{-j2\pi k_0 u/K}$

Table 11.1 summarizes the Fourier descriptors for a boundary sequence $s(k)$ that undergoes rotation, translation, scaling, and changes in starting point. The symbol Δ_{xy} is defined as $\Delta_{xy} = \Delta x + j\Delta y$, so the notation $s_t(k) = s(k) + \Delta_{xy}$ indicates redefining (translating) the sequence as

$$s_t(k) = [x(k) + \Delta x] + j[y(k) + \Delta y]. \tag{11.2-7}$$

In other words, translation consists of adding a constant displacement to all coordinates in the boundary. Note that translation has no effect on the descriptors, except for $u = 0$, which has the impulse function $\delta(u)$.† Finally, the expression $s_p(k) = s(k - k_0)$ means redefining the sequence as

$$s_p = x(k - k_0) + jy(k - k_0), \tag{11.2-8}$$

which merely changes the starting point of the sequence to $k = k_0$ from $k = 0$. The last entry in Table 11.1 shows that a change in starting point affects all descriptors in a different (but known) way, in the sense that the term multiplying $a(u)$ depends on u.

11.2.4 Statistical Moments

The shape of boundary segments (and of signature waveforms) can be described quantitatively by using simple statistical moments, such as the mean, variance, and higher-order moments. To see how this can be accomplished, consider Fig. 11.15(a), which shows the segment of a boundary, and Fig. 11.15(b), which shows the segment represented as a 1-D function $g(r)$ of an arbitrary variable r. This function is obtained by connecting the two end points of the segment and rotating the line segment until it is horizontal. The coordinates of the points are rotated by the same angle.

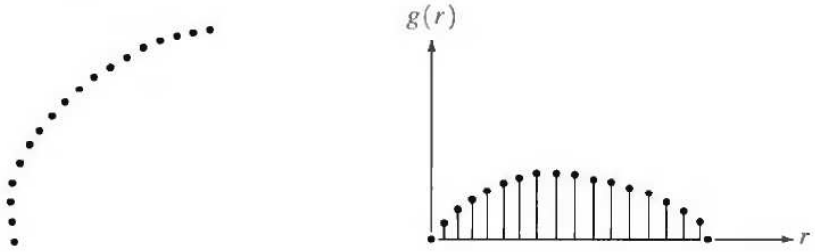
See inside front cover

Consult the book web site for a brief review of probability theory.

Let us treat the amplitude of g as a discrete random variable v and form an amplitude histogram $p(v_i), i = 0, 1, 2, \ldots, A - 1$, where A is the number of discrete amplitude increments in which we divide the amplitude scale. Then, keeping in mind that $p(v_i)$ is an estimate of the probability of value v_i occurring, it follows from Eq. (3.3-18) that the nth moment of v about its mean is

$$\mu_n(v) = \sum_{i=0}^{A-1} (v_i - m)^n p(v_i) \tag{11.2-9}$$

†Recall from Chapter 4 that the Fourier transform of a constant is an impulse located at the origin. Recall also that the impulse function is zero everywhere else.



a b

FIGURE 11.15 (a) Boundary segment. (b) Representation as a 1-D function.

where

$$m = \sum_{i=0}^{A-1} v_i p(v_i). \tag{11.2-10}$$

The quantity m is recognized as the mean or average value of v and μ_2 as its variance. Generally, only the first few moments are required to differentiate between signatures of clearly distinct shapes.

An alternative approach is to normalize $g(r)$ to unit area and treat it as a histogram. In other words, $g(r_i)$ is now treated as the probability of value r_i occurring. In this case, r is treated as the random variable and the moments are

$$\mu_n(r) = \sum_{i=0}^{K-1} (r_i - m)^n g(r_i) \tag{11.2-11}$$

where

$$m = \sum_{i=0}^{K-1} r_i g(r_i). \tag{11.2-12}$$

In this notation, K is the number of points on the boundary, and $\mu_n(r)$ is directly related to the shape of $g(r)$. For example, the second moment $\mu_2(r)$ measures the spread of the curve about the mean value of r and the third moment $\mu_3(r)$ measures its symmetry with reference to the mean.

Basically, what we have accomplished is to reduce the description task to that of describing 1-D functions. Although moments are by far the most popular method, they are not the only descriptors that could be used for this purpose. For instance, another method involves computing the 1-D discrete Fourier transform, obtaining its spectrum, and using the first q components of the spectrum to describe $g(r)$. The advantage of moments over other techniques is that implementation of moments is straightforward and they also carry a "physical" interpretation of boundary shape. The insensitivity of this approach to rotation is clear from Fig. 11.15. Size normalization, if desired, can be achieved by scaling the range of values of g and r.

11.3 Regional Descriptors

In this section we consider various approaches for describing image regions. Keep in mind that it is common practice to use of both boundary and regional descriptors combined.

11.3.1 Some Simple Descriptors

The *area* of a region is defined as the number of pixels in the region. The *perimeter* of a region is the length of its boundary. Although area and perimeter are sometimes used as descriptors, they apply primarily to situations in which the size of the regions of interest is invariant. A more frequent use of these two descriptors is in measuring *compactness* of a region, defined as $(\text{perimeter})^2/\text{area}$. Compactness is a dimensionless quantity (and thus is insensitive to uniform scale changes) and is minimal for a disk-shaped region. With the exception of errors introduced by rotation of a digital region, compactness also is insensitive to orientation.

Other simple measures used as region descriptors include the mean and median of the gray levels, the minimum and maximum gray-level values, and the number of pixels with values above and below the mean.

EXAMPLE 11.4: Using area computations to extract information from images.

■ Even a simple region descriptor such as normalized area can be quite useful in extracting information from images. For instance, Fig. 11.16 shows a satellite infrared image of the Americas. As discussed in more detail in Section 1.3.4, images such as these provide a global inventory of human settlements. The sensor used to collect these images has the capability to detect visible and near-infrared emissions, such as lights, fires, and flares. The table alongside the images shows (by region from top to bottom) the ratio of the area occupied by white (the lights) to the total light area in all four regions. A simple measurement like this can give, for example, a relative estimate by region of electrical energy consumed. The data can be refined by normalizing it with respect to land mass per region, with respect to population numbers, and so on. ■

11.3.2 Topological Descriptors

Topological properties are useful for global descriptions of regions in the image plane. Simply defined, *topology* is the study of properties of a figure that are unaffected by any deformation, as long as there is no tearing or joining of the figure (sometimes these are called *rubber-sheet* distortions). For example, Fig. 11.17 shows a region with two holes. Thus if a topological descriptor is defined by the number of holes in the region, this property obviously will not be affected by a stretching or rotation transformation. In general, however, the number of holes will change if the region is torn or folded. Note that, as stretching affects distance, topological properties do not depend on the notion of distance or any properties implicitly based on the concept of a distance measure.

Another topological property useful for region description is the number of connected components. A *connected component* of a region was defined in Section 2.5.2. Figure 11.18 shows a region with three connected components. (See Section 9.5.3 regarding an algorithm for computing connected components.)

The number of holes H and connected components C in a figure can be used to define the *Euler number* E:

$$E = C - H \tag{11.3-1}$$

Region no, (from top)	Ratio of lights per region to total lights
1	0.204
2	0.640
3	0.049
4	0.107

FIGURE 11.16 Infrared images of the Americas at night. (Courtesy of NOAA.)

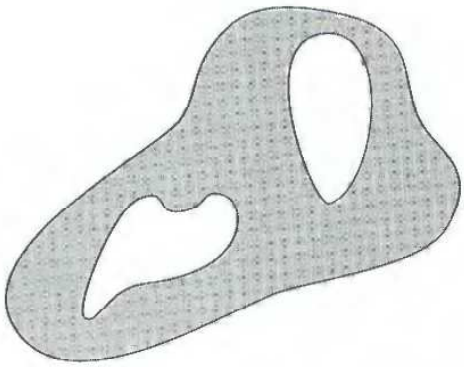


FIGURE 11.17 A region with two holes.

The Euler number is also a topological property. The regions shown in Fig. 11.19, for example, have Euler numbers equal to 0 and −1, respectively, because the "A" has one connected component and one hole and the "B" one connected component but two holes.

Regions represented by straight-line segments (referred to as *polygonal networks*) have a particularly simple interpretation in terms of the Euler number. Figure 11.20 shows a polygonal network. Classifying interior regions of such a network into faces and holes often is important. Denoting the number of vertices by V, the number of edges by Q, and the number of faces by F gives the following relationship, called the *Euler formula*:

$$V - Q + F = C - H \tag{11.3-2}$$

which, in view of Eq. (11.3-1), is equal to the Euler number:

$$\begin{aligned} V - Q + F &= C - H \\ &= E. \end{aligned} \tag{11.3-3}$$

The network shown in Fig. 11.20 has 7 vertices, 11 edges, 2 faces, 1 connected region, and 3 holes; thus the Euler number is −2:

$$7 - 11 + 2 = 1 - 3 = -2.$$

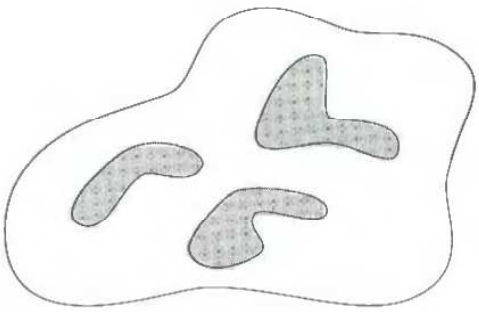
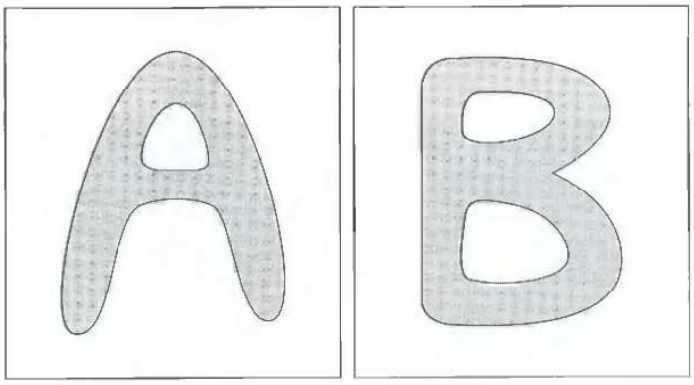


FIGURE 11.18 A region with three connected components.



a b

FIGURE 11.19 Regions with Euler number equal to 0 and −1, respectively.

Topological descriptors provide an additional feature that is often useful in characterizing regions in a scene.

EXAMPLE 11.5: Use of connected components for extracting the largest features in a segmented image.

■ Figure 11.21(a) shows a 512 × 512, 8-bit image of Washington, D.C. taken by a NASA LANDSAT satellite. This particular image is in the near infrared band (see Fig. 1.10 for details). Suppose that we want to segment the river using only this image (as opposed to using several multispectral images, which would simplify the task). Since the river is a rather dark, uniform region of the image, thresholding is an obvious thing to try. The result of thresholding the image with the highest possible threshold value before the river became a disconnected region is shown in Fig. 11.21(b). The threshold was selected manually to illustrate the point that it would be impossible in this case to segment the river by itself without other regions of the image also appearing in the thresholded result. The objective of this example is to illustrate how connected components can be used to "finish" the segmentation.

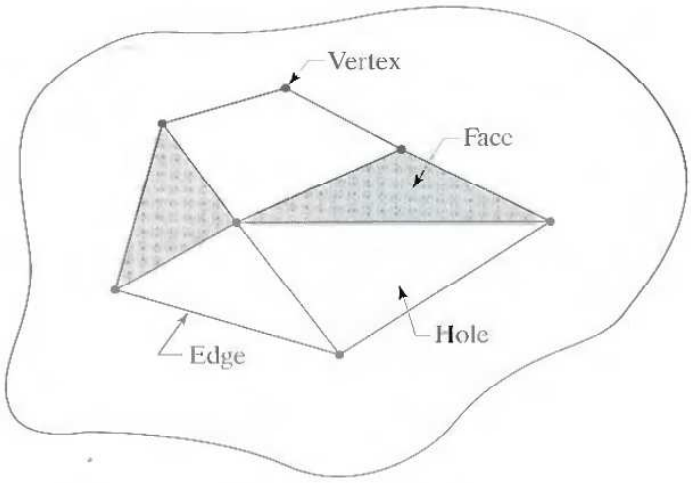
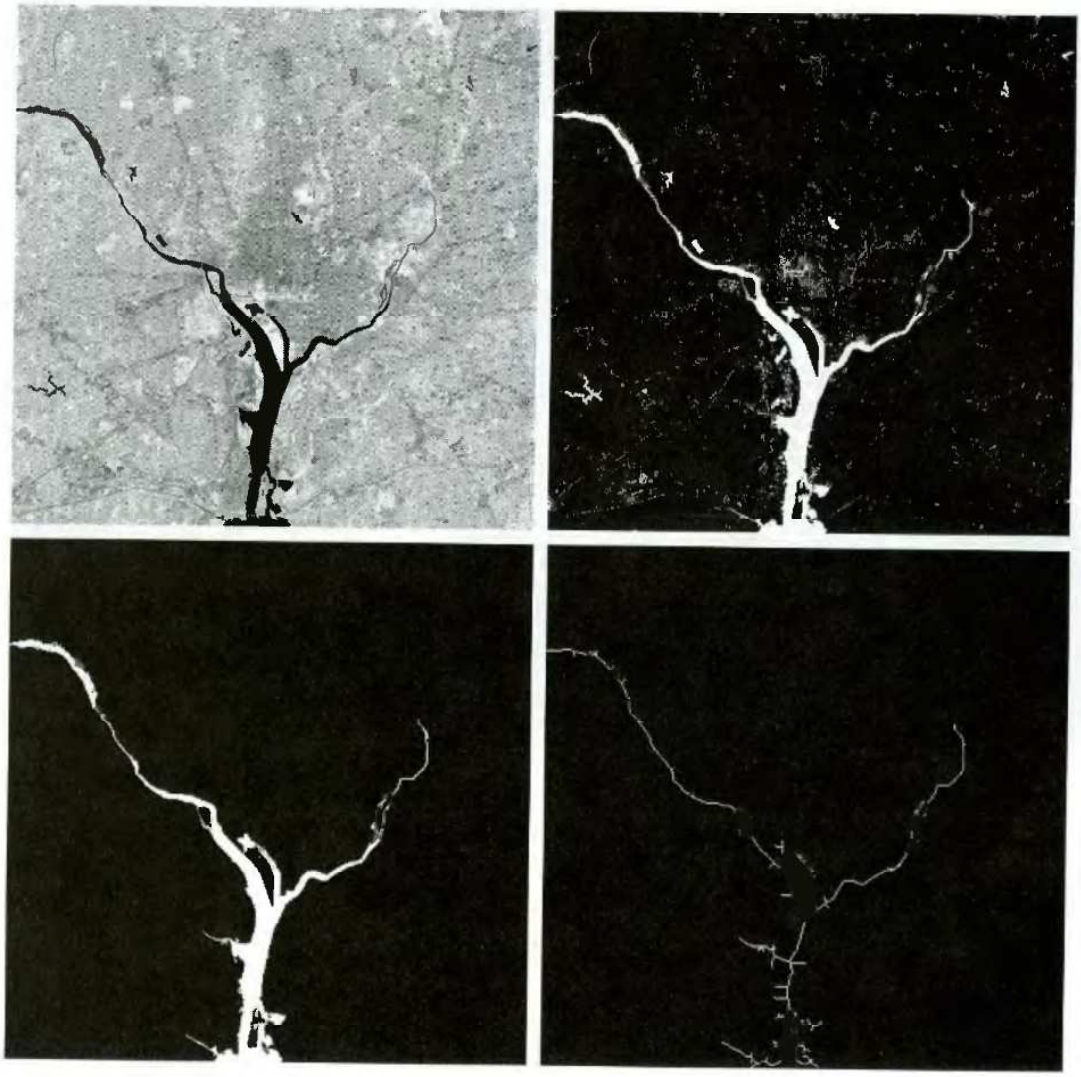


FIGURE 11.20 A region containing a polygonal network.



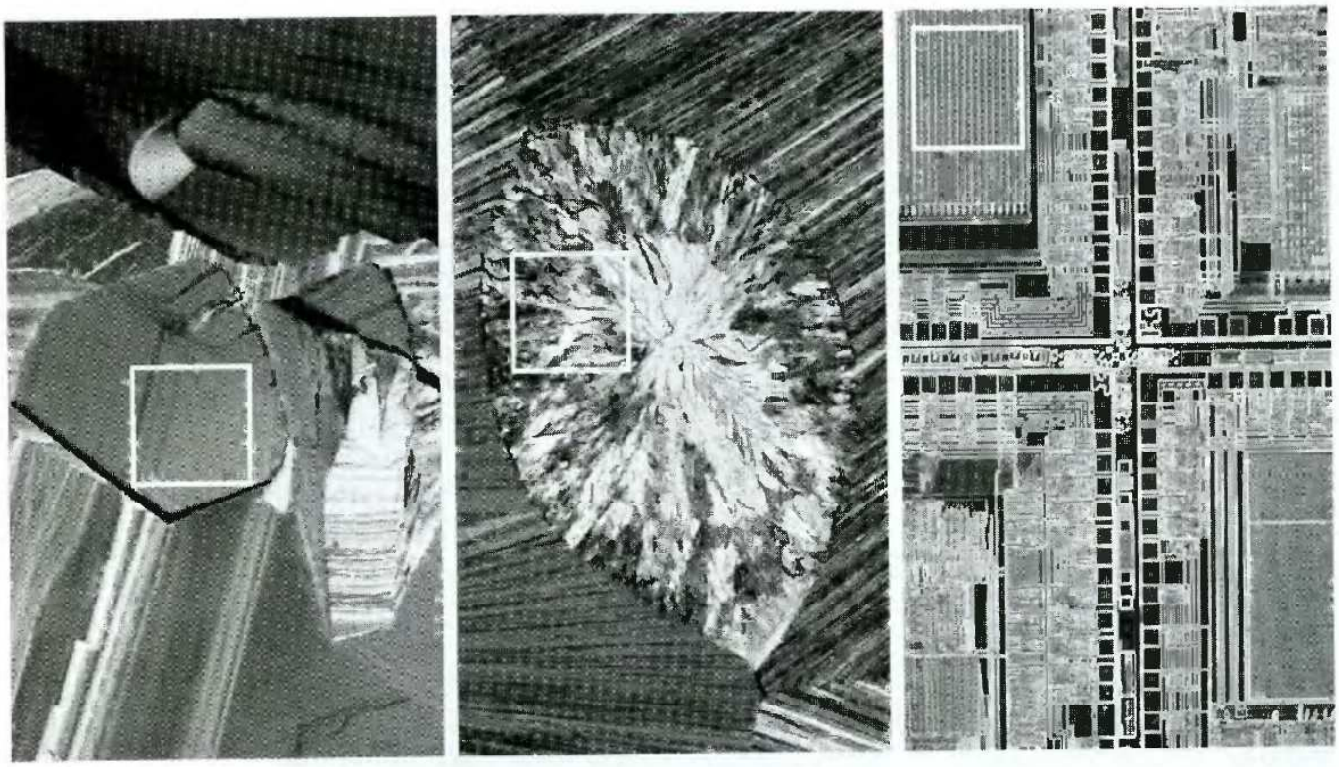
a b
c d

FIGURE 11.21 (a) Infrared image of the Washington, D.C. area. (b) Thresholded image. (c) The largest connected component of (b). Skeleton of (c).

The image in Fig. 11.21(b) has 1591 connected components (obtained using 8-connectivity) and its Euler number is 1552, from which we deduce that the number of holes is 39. Figure 11.21(c) shows the connected component with the largest number of elements (8479). This is the desired result, which we already know cannot be segmented by itself from the image. Note how clean this result is. If we wanted to perform measurements, like the length of each branch of the river, we could use the skeleton of the connected component [Fig. 11.21(d)] to do so. In other words, the length of each branch in the skeleton would be a reasonably close approximation to the length of the river branch it represents. ■

11.3.3 Texture

An important approach to region description is to quantify its *texture* content. Although no formal definition of texture exists, intuitively this descriptor provides measures of properties such as smoothness, coarseness, and regularity (Fig. 11.22 shows some examples). The three principal approaches used in image processing to describe the texture of a region are statistical, structural, and spectral. Statistical approaches yield characterizations of textures as smooth, coarse,



a b c

FIGURE 11.22 The white squares mark, from left to right, smooth, coarse, and regular textures. These are optical microscope images of a superconductor, human cholesterol, and a microprocessor. (Courtesy of Dr. Michael W. Davidson, Florida State University.)

grainy, and so on. Structural techniques deal with the arrangement of image primitives, such as the description of texture based on regularly spaced parallel lines. Spectral techniques are based on properties of the Fourier spectrum and are used primarily to detect global periodicity in an image by identifying high-energy, narrow peaks in the spectrum.

Statistical approaches

One of the simplest approaches for describing texture is to use statistical moments of the gray-level histogram of an image or region. Let z be a random variable denoting gray levels and let $p(z_i), i = 0, 1, 2, \ldots, L - 1$, be the corresponding histogram, where L is the number of distinct gray levels. From Eq. (3.3-18), the nth moment of z about the mean is

$$\mu_n(z) = \sum_{i=0}^{L-1} (z_i - m)^n p(z_i) \tag{11.3-4}$$

where m is the mean value of z (the average gray level):

$$m = \sum_{i=0}^{L-1} z_i p(z_i). \tag{11.3-5}$$

Note from Eq. (11.3-4) that $\mu_0 = 1$ and $\mu_1 = 0$. The second moment [the *variance* $\sigma^2(z) = \mu_2(z)$] is of particular importance in texture description. It is a measure of gray-level contrast that can be used to establish descriptors of relative smoothness. For example, the measure

$$R = 1 - \frac{1}{1 + \sigma^2(z)} \tag{11.3-6}$$

is 0 for areas of constant intensity (the variance is zero there) and approaches 1 for large values of $\sigma^2(z)$. Because variance values tend to be large for gray-scale images with values, for example, in the range 0 to 255, it is a good idea to normalize the variance to the interval [0, 1] for use in Eq. (11.3-6). This is done simply by dividing $\sigma^2(z)$ by $(L - 1)^2$ in Eq. (11.3-6). The standard deviation, $\sigma(z)$, also is used frequently as a measure of texture because values of the standard deviation tend to be more intuitive to many people.

The third moment,

$$\mu_3(z) = \sum_{i=0}^{L-1} (z_i - m)^3 p(z_i), \tag{11.3-7}$$

is a measure of the skewness of the histogram while the fourth moment is a measure of its relative flatness. The fifth and higher moments are not so easily related to histogram shape, but they do provide further quantitative discrimination of texture content. Some useful additional texture measures based on histograms include a measure of "uniformity," given by

$$U = \sum_{i=0}^{L-1} p^2(z_i), \tag{11.3-8}$$

and an *average entropy* measure, which the reader might recall from basic information theory, or from our discussion in Chapter 8, is defined as

$$e = -\sum_{i=0}^{L-1} p(z_i) \log_2 p(z_i). \tag{11.3-9}$$

Because the p's have values in the range [0, 1] and their sum equals 1, measure U is maximum for an image in which all gray levels are equal (maximally uniform), and decreases from there. Entropy is a measure of variability and is 0 for a constant image.

EXAMPLE 11.6: Texture measures based on histograms.

■ Table 11.2 summarizes the values of the preceding measures for the three types of textures highlighted in Fig. 11.22. The mean just tells us the average gray level of each region and is useful only as a rough idea of intensity, not really texture. The standard deviation is much more informative; the numbers clearly show that the first texture has significantly less variability in gray level (it is smoother) than the other two textures. The coarse texture shows up clearly in this measure. As expected, the same comments hold for R, because it measures essentially the same thing as the standard deviation. The third moment generally is useful for determining the degree of symmetry of histograms and whether they are skewed to the left (negative value) or the right (positive value).

TABLE 11.2
Texture measures for the subimages shown in Fig. 11.22.

Texture	Mean	Standard deviation	R (normalized)	Third moment	Uniformity	Entropy
Smooth	82.64	11.79	0.002	−0.105	0.026	5.434
Coarse	143.56	74.63	0.079	−0.151	0.005	7.783
Regular	99.72	33.73	0.017	0.750	0.013	6.674

This gives a rough idea of whether the gray levels are biased toward the dark or light side of the mean. In terms of texture, the information derived from the third moment is useful only when variations between measurements are large. Looking at the measure of uniformity, we again conclude that the first subimage is smoother (more uniform than the rest) and that the most random (lowest uniformity) corresponds to the coarse texture. This is not surprising. Finally, the entropy values are in the opposite order and thus lead us to the same conclusions as the uniformity measure did. The first subimage has the lowest variation in gray level and the coarse image the most. The regular texture is in between the two extremes with respect to both these measures. ■

Measures of texture computed using only histograms suffer from the limitation that they carry no information regarding the relative position of pixels with respect to each other. One way to bring this type of information into the texture-analysis process is to consider not only the distribution of intensities, but also the positions of pixels with equal or nearly equal intensity values.

Let P be a position operator and let $\mathbf{A}$ be a $k \times k$ matrix whose element a_{ij} is the number of times that points with gray level z_i occur (in the position specified by P) relative to points with gray level z_j, with $1 \leq i, j \leq k$. For instance, consider an image with three gray levels, $z_1 = 0$, $z_2 = 1$, and $z_3 = 2$, as follows:

$$\begin{matrix} 0 & 0 & 0 & 1 & 2 \\ 1 & 1 & 0 & 1 & 1 \\ 2 & 2 & 1 & 0 & 0 \\ 1 & 1 & 0 & 2 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{matrix}$$

Defining the position operator P as "one pixel to the right and one pixel below" yields the following 3×3 matrix $\mathbf{A}$:

$$\mathbf{A} = \begin{bmatrix} 4 & 2 & 1 \\ 2 & 3 & 2 \\ 0 & 2 & 0 \end{bmatrix}$$

where, for example, a_{11} (top left) is the number of times that a point with level $z_1 = 0$ appears one pixel location below and to the right of a pixel with the same gray level, and a_{13} (top right) is the number of times that a point with level $z_1 = 0$ appears one pixel location below and to the right of a point with gray level $z_3 = 2$. The size of $\mathbf{A}$ is determined by the number of distinct gray levels in the input image. Thus application of the concepts discussed in this section

usually requires that intensities be requantized into a few gray-level bands in order to keep the size of **A** manageable.

Let n be the total number of point pairs in the image that satisfy P (in the preceding example $n = 16$, the sum of all values in matrix **A**). If a matrix **C** is formed by dividing every element of **A** by n, then c_{ij} is an estimate of the joint probability that a pair of points satisfying P will have values (z_i, z_j). The matrix **C** is called the *gray-level co-occurrence matrix*. Because **C** depends on P, the presence of given texture patterns may be detected by choosing an appropriate position operator. For instance, the operator used in the preceding example is sensitive to bands of constant intensity running at −45°. (Note that the highest value in **A** was $a_{11} = 4$, partially due to a streak of points with intensity 0 and running at −45°.) More generally, the problem is to analyze a given **C** matrix in order to categorize the texture of the region over which **C** was computed. A set of descriptors useful for this purpose includes the following:

1. Maximum probability

$$\max_{i,j}(c_{ij})$$

2. Element difference moment of order k

$$\sum_i \sum_j (i - j)^k c_{ij}$$

3. Inverse element difference moment of order k

$$\sum_i \sum_j c_{ij}/(i - j)^k \qquad i \neq j$$

4. Uniformity

$$\sum_i \sum_j c_{ij}^2$$

5. Entropy

$$-\sum_i \sum_j c_{ij} \log_2 c_{ij}$$

The basic idea is to characterize the "content" of **C** via these descriptors. For example, the first property gives an indication of the strongest response to P. The second descriptor has a relatively low value when the high values of **C** are near the main diagonal, because the differences $(i - j)$ are smaller there. The third descriptor has the opposite effect. The fourth descriptor is highest when the c_{ij}s are all equal. As noted previously, the fifth descriptor is a measure of randomness, achieving its highest value when all elements of **C** are maximally random.

One approach for using these descriptors is to "teach" a system representative descriptor values for a set of different textures. The texture of an unknown region is then subsequently determined by how closely its descriptors match those stored in the system memory. We discuss matching in more detail in Chapter 12.

Structural approaches

As mentioned at the beginning of this section, a second major category of texture description is based on structural concepts. Suppose that we have a rule of the form $S \rightarrow aS$, which indicates that the symbol S may be rewritten as aS (for example, three applications of this rule would yield the string $aaaS$). If a represents a circle [Fig. 11.23(a)] and the meaning of "circles to the right" is assigned to a string of the form $aaa\ldots$, the rule $S \rightarrow aS$ allows generation of the texture pattern shown in Fig. 11.23(b).

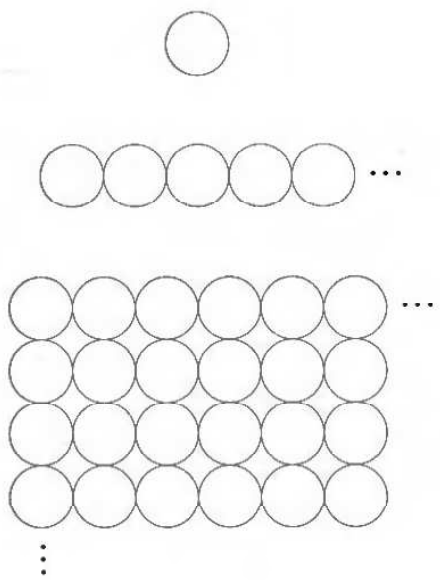
Suppose next that we add some new rules to this scheme: $S \rightarrow bA$, $A \rightarrow cA$, $A \rightarrow c$, $A \rightarrow bS$, $S \rightarrow a$, where the presence of a b means "circle down" and the presence of a c means "circle to the left." We can now generate a string of the form $aaabccbaa$ that corresponds to a 3×3 matrix of circles. Larger texture patterns, such as the one shown in Fig. 11.23(c), can be generated easily in the same way. (Note, however, that these rules can also generate structures that are not rectangular.)

The basic idea in the foregoing discussion is that a simple "texture primitive" can be used to form more complex texture patterns by means of some rules that limit the number of possible arrangements of the primitive(s). These concepts lie at the heart of relational descriptions, a topic that we treat in more detail in Section 11.5.

Spectral approaches

As indicated in Section 5.4, the Fourier spectrum is ideally suited for describing the directionality of periodic or almost periodic 2-D patterns in an image. These global texture patterns, although easily distinguishable as concentrations of high-energy bursts in the spectrum, generally are quite difficult to detect with spatial methods because of the local nature of these techniques.

Here, we consider three features of the Fourier spectrum that are useful for texture description: (1) Prominent peaks in the spectrum give the principal



a
b
c

FIGURE 11.23 (a) Texture primitive. (b) Pattern generated by the rule $S \rightarrow aS$. (c) 2-D texture pattern generated by this and other rules.

direction of the texture patterns. (2) The location of the peaks in the frequency plane gives the fundamental spatial period of the patterns. (3) Eliminating any periodic components via filtering leaves nonperiodic image elements, which can then be described by statistical techniques. Recall that the spectrum is symmetric about the origin, so only half of the frequency plane needs to be considered. Thus for the purpose of analysis, every periodic pattern is associated with only one peak in the spectrum, rather than two.

Detection and interpretation of the spectrum features just mentioned often are simplified by expressing the spectrum in polar coordinates to yield a function $S(r, \theta)$, where S is the spectrum function and r and θ are the variables in this coordinate system. For each direction θ, $S(r, \theta)$ may be considered a 1-D function $S_\theta(r)$. Similarly, for each frequency r, $S_r(\theta)$ is a 1-D function. Analyzing $S_\theta(r)$ for a fixed value of θ yields the behavior of the spectrum (such as the presence of peaks) along a radial direction from the origin, whereas analyzing $S_r(\theta)$ for a fixed value of r yields the behavior along a circle centered on the origin.

A more global description is obtained by integrating (summing for discrete variables) these functions:

$$S(r) = \sum_{\theta=0}^{\pi} S_\theta(r) \tag{11.3-10}$$

and

$$S(\theta) = \sum_{r=1}^{R_0} S_r(\theta) \tag{11.3-11}$$

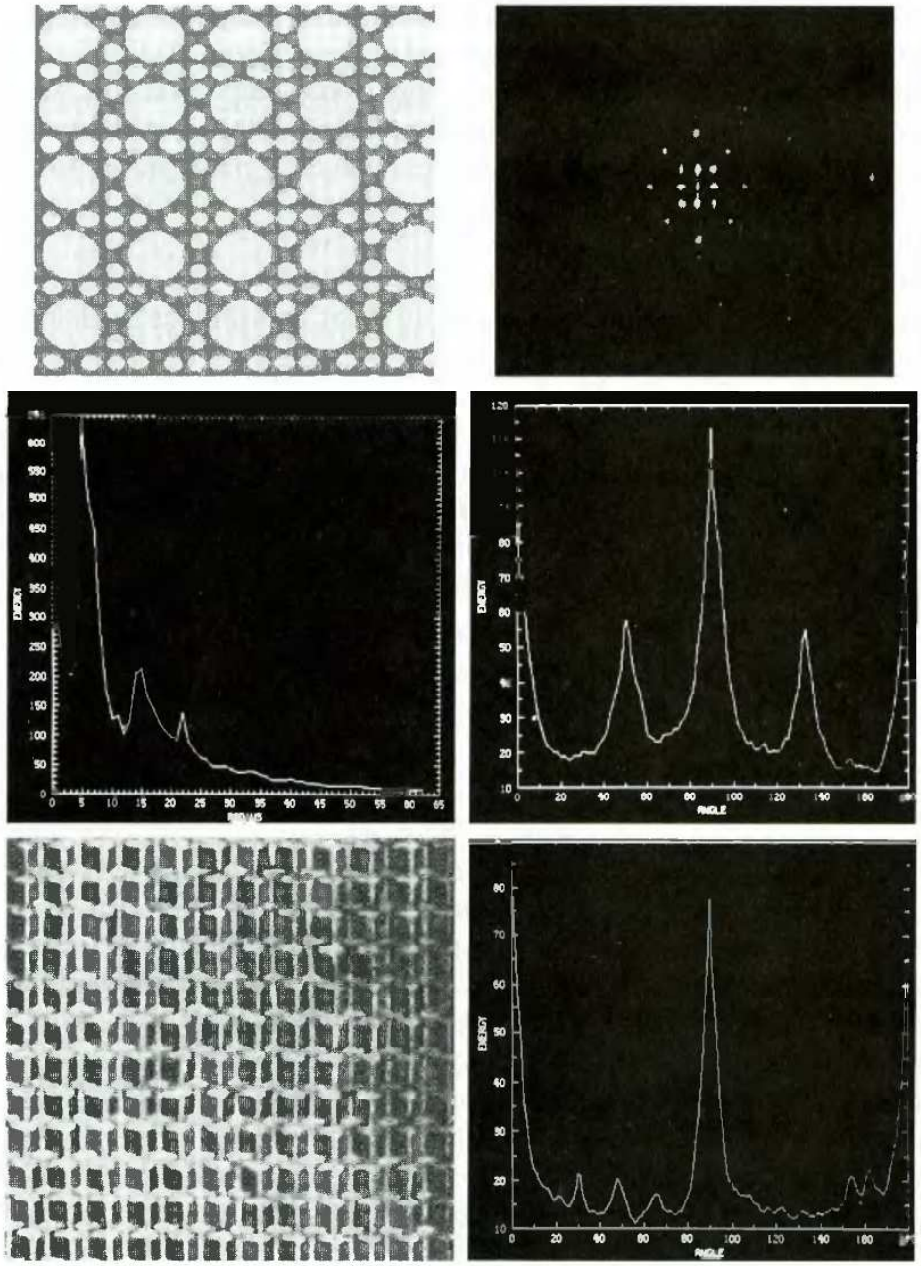
where R_0 is the radius of a circle centered at the origin.

The results of Eqs. (11.3-10) and (11.3-11) constitute a pair of values $[S(r), S(\theta)]$ for *each* pair of coordinates (r, θ). By varying these coordinates, we can generate two 1-D functions, $S(r)$ and $S(\theta)$, that constitute a spectral-energy description of texture for an entire image or region under consideration. Furthermore, descriptors of these functions themselves can be computed in order to characterize their behavior quantitatively. Descriptors typically used for this purpose are the location of the highest value, the mean and variance of both the amplitude and axial variations, and the distance between the mean and the highest value of the function.

EXAMPLE 11.7: Spectral texture.

■ Figure 11.24 illustrates the use of Eqs. (11.3-10) and (11.3-11) for global texture description. Figure 11.24(a) shows an image with periodic texture, and Fig. 11.24(b) shows its spectrum. Figures 11.24(c) and (d) show plots of $S(r)$ and $S(\theta)$, respectively. The plot of $S(r)$ is a typical structure, having high energy content near the origin and progressively lower values for higher frequencies. The plot of $S(\theta)$ shows prominent peaks at intervals of 45°, which clearly correspond to the periodicity in the texture content of the image.

As an illustration of how a plot of $S(\theta)$ could be used to differentiate between two texture patterns, Fig. 11.24(e) shows another image whose texture pattern is predominantly in the horizontal and vertical directions. Figure 11.24(f) shows the plot of $S(\theta)$ for the spectrum of this image. As expected, this plot shows peaks at 90° intervals. Discriminating between the two texture patterns by analyzing their corresponding $S(\theta)$ waveforms would be straightforward. ■



a b
c d
e f

FIGURE 11.24 (a) Image showing periodic texture. (b) Spectrum. (c) Plot of $S(r)$. (d) Plot of $S(\theta)$. (e) Another image with a different type of periodic texture. (f) Plot of $S(\theta)$. (Courtesy of Dr. Dragana Brzakovic, University of Tennessee.)

11.3.4 Moments of Two-Dimensional Functions

For a 2-D continuous function $f(x, y)$, the moment of order $(p + q)$ is defined as

$$m_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^p y^q f(x, y)\, dx\, dy \tag{11.3-12}$$

for $p, q = 0, 1, 2, \ldots$. A uniqueness theorem (Papoulis [1991]) states that if $f(x, y)$ is piecewise continuous and has nonzero values only in a finite part of

the xy-plane, moments of all orders exist, and the moment sequence (m_{pq}) is uniquely determined by $f(x, y)$. Conversely, (m_{pq}) uniquely determines $f(x, y)$.

The *central moments* are defined as

$$\mu_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(x - \bar{x})^p(y - \bar{y})^q f(x, y)\, dx\, dy \qquad (11.3\text{-}13)$$

where

$$\bar{x} = \frac{m_{10}}{m_{00}} \quad \text{and} \quad \bar{y} = \frac{m_{01}}{m_{00}}.$$

If $f(x, y)$ is a digital image, then Eq. (11.3-13) becomes

$$\mu_{pq} = \sum_x \sum_y (x - \bar{x})^p(y - \bar{y})^q f(x, y). \qquad (11.3\text{-}14)$$

The central moments of order up to 3 are

$$\begin{aligned}
\mu_{00} &= \sum_x \sum_y (x - \bar{x})^0(y - \bar{y})^0 f(x, y) \\
&= \sum_x \sum_y f(x, y) \\
&= m_{00} \\
\mu_{10} &= \sum_x \sum_y (x - \bar{x})^1(y - \bar{y})^0 f(x, y) \\
&= m_{10} - \frac{m_{10}}{m_{00}}(m_{00}) \\
&= 0 \\
\mu_{01} &= \sum_x \sum_y (x - \bar{x})^0(y - \bar{y})^1 f(x, y) \\
&= m_{01} - \frac{m_{01}}{m_{00}}(m_{00}) \\
&= 0 \\
\mu_{11} &= \sum_x \sum_y (x - \bar{x})^1(y - \bar{y})^1 f(x, y) \\
&= m_{11} - \frac{m_{10}m_{01}}{m_{00}} \\
&= m_{11} - \bar{x}m_{01} = m_{11} - \bar{y}m_{10} \\
\mu_{20} &= \sum_x \sum_y (x - \bar{x})^2(y - \bar{y})^0 f(x, y) \\
&= m_{20} - \frac{2m_{10}^2}{m_{00}} + \frac{m_{10}^2}{m_{00}} \\
&= m_{20} - \frac{m_{10}^2}{m_{00}} \\
&= m_{20} - \bar{x}m_{10}
\end{aligned}$$

$$\begin{aligned}
\mu_{02} &= \sum_x \sum_y (x - \bar{x})^0 (y - \bar{y})^2 f(x, y) \\
&= m_{02} - \frac{m_{01}^2}{m_{00}} \\
&= m_{02} - \bar{y} m_{01} \\
\mu_{21} &= \sum_x \sum_y (x - \bar{x})^2 (y - \bar{y})^1 f(x, y) \\
&= m_{21} - 2\bar{x} m_{11} - \bar{y} m_{20} + 2\bar{x}^2 m_{01} \\
\mu_{12} &= \sum_x \sum_y (x - \bar{x})^1 (y - \bar{y})^2 f(x, y) \\
&= m_{12} - 2\bar{y} m_{11} - \bar{x} m_{02} + 2\bar{y}^2 m_{10} \\
\mu_{30} &= \sum_x \sum_y (x - \bar{x})^3 (y - \bar{y})^0 f(x, y) \\
&= m_{30} - 3\bar{x} m_{20} + 2\bar{x}^2 m_{10} \\
\mu_{03} &= \sum_x \sum_y (x - \bar{x})^0 (y - \bar{y})^3 f(x, y) \\
&= m_{03} - 3\bar{y} m_{02} + 2\bar{y}^2 m_{01}.
\end{aligned}$$

In summary,

$$\begin{aligned}
\mu_{00} &= m_{00} & \mu_{02} &= m_{02} - \bar{y} m_{01} \\
\mu_{10} &= 0 & \mu_{30} &= m_{30} - 3\bar{x} m_{20} + 2\bar{x}^2 m_{10} \\
\mu_{01} &= 0 & \mu_{03} &= m_{03} - 3\bar{y} m_{02} + 2\bar{y}^2 m_{01} \\
\mu_{11} &= m_{11} - \bar{y} m_{10} & \mu_{21} &= m_{21} - 2\bar{x} m_{11} - \bar{y} m_{20} + 2\bar{x}^2 m_{01} \\
\mu_{20} &= m_{20} - \bar{x} m_{10} & \mu_{12} &= m_{12} - 2\bar{y} m_{11} - \bar{x} m_{02} + 2\bar{y}^2 m_{10}.
\end{aligned}$$

The *normalized central moments*, denoted η_{pq}, are defined as

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}} \tag{11.3-15}$$

where

$$\gamma = \frac{p + q}{2} + 1 \tag{11.3-16}$$

for $p + q = 2, 3, \ldots$.

A set of seven *invariant moments* can be derived from the second and third moments.†

$$\phi_1 = \eta_{20} + \eta_{02} \tag{11.3-17}$$

$$\phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \tag{11.3-18}$$

† Derivation of these results involves concepts that are beyond the scope of this discussion. The book by Bell [1965] and the paper by Hu [1962] contain detailed discussions of these concepts. Moment invariants can be generalized to n dimensions (Mamistvalov [1998]).

$$\phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \tag{11.3-19}$$

$$\phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \tag{11.3-20}$$

$$\begin{aligned}\phi_5 = {} & (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 \\ & - 3(\eta_{21} + \eta_{03})^2] + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03}) \\ & [3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]\end{aligned} \tag{11.3-21}$$

$$\begin{aligned}\phi_6 = {} & (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \\ & + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})\end{aligned} \tag{11.3-22}$$

$$\begin{aligned}\phi_7 = {} & (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 \\ & - 3(\eta_{21} + \eta_{03})^2] + (3\eta_{12} - \eta_{30})(\eta_{21} + \eta_{03}) \\ & [3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2].\end{aligned} \tag{11.3-23}$$

This set of moments is invariant to translation, rotation, and scale change.

EXAMPLE 11.8: Two-dimensional moment invariants.

The image shown in Fig. 11.25(a) was reduced to half size in Fig. 11.25(b), mirror-imaged in Fig. 11.25(c), and rotated by 2° and 45°, as shown in Figs. 11.25(d) and (e). The seven moment invariants given in Eqs. (11.3-17) through (11.3-23) were then computed for each of these images, and the logarithm of the results were taken to reduce the dynamic range. As Table 11.3 shows, the results for Figs. 11.25(b) through (e) are in reasonable agreement with the invariants computed for the original image. The major cause of error can be attributed to the digital nature of the data, especially for the rotated images.

11.4 Use of Principal Components for Description

See inside front cover

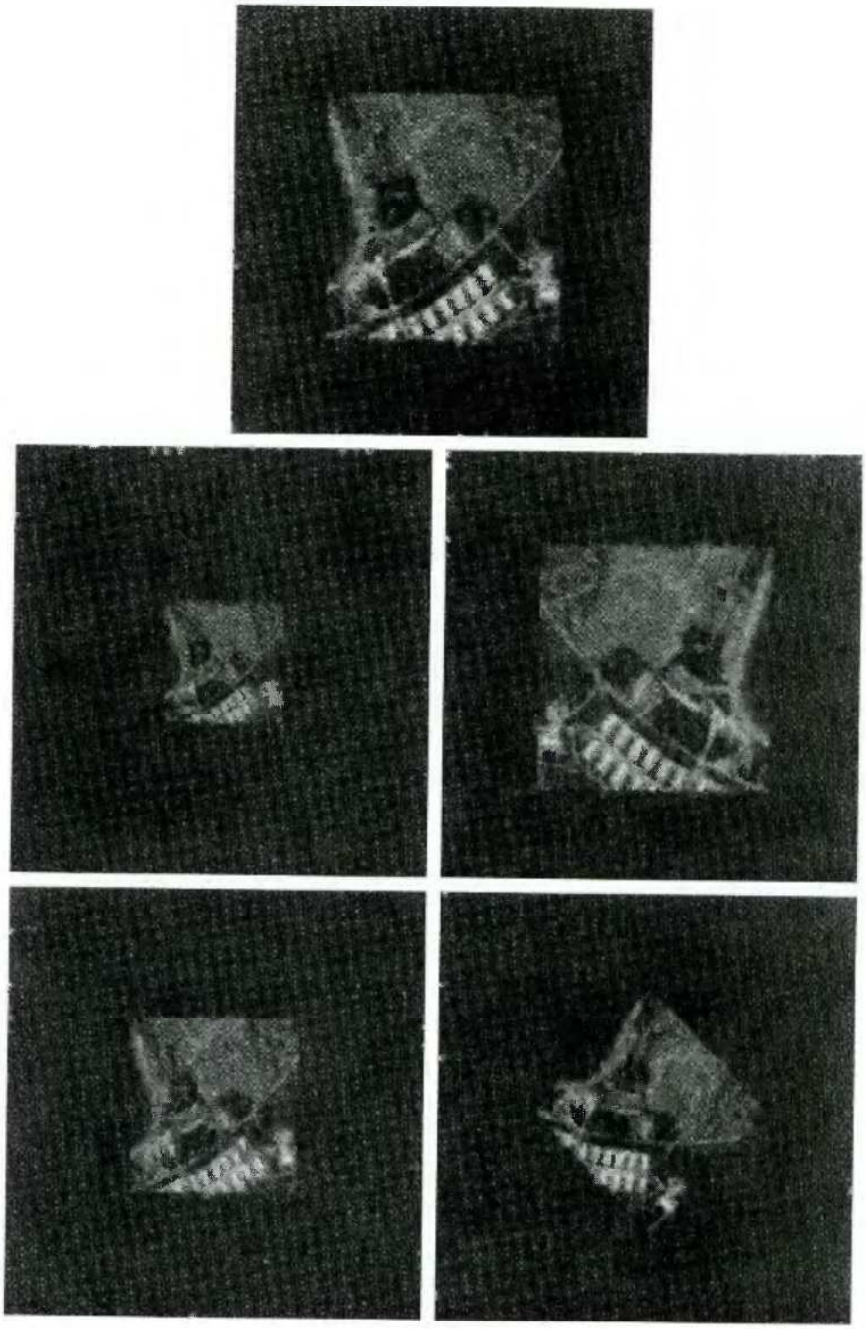
Consult the book web site for a brief review of vectors and matrices.

The material discussed in this section is applicable to boundaries and regions. In addition, it can be used as the basis for describing sets of images that are registered spatially, but whose corresponding pixel values are different (e.g., the three component images of a color RGB image). Suppose that we are given the three component images of such a color image. The three images can be treated as a unit by expressing each group of three corresponding pixels as a vector. For example, let x_1, x_2, and x_3, respectively, be the values of the first pixel in each of the three images. These three elements can be expressed in the form of a 3-D *column* vector, $\mathbf{x}$, where

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}.$$

This one vector represents *one* common pixel in all three images. If the images are of size $M \times N$, there will be a total of $K = MN$ three-dimensional vectors after all the pixels are represented in this manner. If we have n registered images, the vectors will be n-dimensional:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}. \tag{11.4-1}$$



a
b c
d e

FIGURE 11.25 Images used to demonstrate properties of moment invariants (see Table 11.3).

Throughout this section, the assumption is that all vectors are column vectors (i.e., matrices of order $n \times 1$). We can write them on a line of text simply by expressing them as $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$, where "$T$" indicates transpose.

We can treat the vectors as random quantities, just like we did when constructing a gray-level histogram. The only difference is that, instead of talking about quantities like the mean and variance of the random variables, we now talk about *mean vectors* and *covariance matrices* of the random vectors. The mean vector of the population is defined as

$$\mathbf{m}_{\mathbf{x}} = E\{\mathbf{x}\} \tag{11.4-2}$$

Invariant (Log)	Original	Half Size	Mirrored	Rotated 2°	Rotated 45°
ϕ_1	6.249	6.226	6.919	6.253	6.318
ϕ_2	17.180	16.954	19.955	17.270	16.803
ϕ_3	22.655	23.531	26.689	22.836	19.724
ϕ_4	22.919	24.236	26.901	23.130	20.437
ϕ_5	45.749	48.349	53.724	46.136	40.525
ϕ_6	31.830	32.916	37.134	32.068	29.315
ϕ_7	45.589	48.343	53.590	46.017	40.470

TABLE 11.3 Moment invariants for the images in Figs. 11.25(a)-(e).

where $E\{\cdot\}$ is the expected value of the argument, and the subscript denotes that $\mathbf{m}$ is associated with the population of $\mathbf{x}$ vectors. Recall that the expected value of a vector or matrix is obtained by taking the expected value of each element.

The *covariance matrix* of the vector population is defined as

$$\mathbf{C}_\mathbf{x} = E\{(\mathbf{x} - \mathbf{m}_\mathbf{x})(\mathbf{x} - \mathbf{m}_\mathbf{x})^T\}. \tag{11.4-3}$$

Because $\mathbf{x}$ is n dimensional, $\mathbf{C}_\mathbf{x}$ and $(\mathbf{x} - \mathbf{m}_\mathbf{x})(\mathbf{x} - \mathbf{m}_\mathbf{x})^T$ are matrices of order $n \times n$. Element c_{ii} of $\mathbf{C}_\mathbf{x}$ is the variance of x_i, the ith component of the $\mathbf{x}$ vectors in the population, and element c_{ij} of $\mathbf{C}_\mathbf{x}$ is the covariance† between elements x_i and x_j of these vectors. The matrix $\mathbf{C}_\mathbf{x}$ is real and symmetric. If elements x_i and x_j are uncorrelated, their covariance is zero and, therefore, $c_{ij} = c_{ji} = 0$. Note that all these definitions reduce to their familiar one-dimensional counterparts when $n = 1$.

For K vector samples from a random population, the mean vector can be approximated from the samples by using the familiar averaging expression

$$\mathbf{m}_\mathbf{x} = \frac{1}{K}\sum_{k=1}^{K}\mathbf{x}_k. \tag{11.4-4}$$

Similarly, by expanding the product $(\mathbf{x} - \mathbf{m}_\mathbf{x})(\mathbf{x} - \mathbf{m}_\mathbf{x})^T$ and using Eqs. (11.4-2) and (11.4-4) we would find that the covariance matrix can be approximated from the samples as follows:

$$\mathbf{C}_\mathbf{x} = \frac{1}{K}\sum_{k=1}^{K}\mathbf{x}_k\mathbf{x}_k^T - \mathbf{m}_\mathbf{x}\mathbf{m}_\mathbf{x}^T. \tag{11.4-5}$$

EXAMPLE 11.9: Computation of the mean vector and covariance matrix.

To illustrate the mechanics of Eqs. (11.4-4) and (11.4-5), consider the four vectors $\mathbf{x}_1 = (0,0,0)^T$, $\mathbf{x}_2 = (1,0,0)^T$, $\mathbf{x}_3 = (1,1,0)^T$, and $\mathbf{x}_4 = (1,0,1)^T$, where the transpose is used so that column vectors may be conveniently written horizontally on a line of text, as noted previously. Applying Eq. (11.4-4) yields the following mean vector:

$$\mathbf{m}_\mathbf{x} = \frac{1}{4}\begin{bmatrix} 3 \\ 1 \\ 1 \end{bmatrix}.$$

† Recall that the variance of a random variable x with mean m is defined as $E\{(x - m)^2\}$. The covariance of two random variables x_i and x_j is defined as $E\{(x_i - m_i)(x_j - m_j)\}$. If the variables are *uncorrelated*, their covariance is 0.

Similarly, use of Eq. (11.4-5) yields the following covariance matrix:

$$\mathbf{C_x} = \frac{1}{16}\begin{bmatrix} 3 & 1 & 1 \\ 1 & 3 & -1 \\ 1 & -1 & 3 \end{bmatrix}.$$

All the elements along the main diagonal are equal, which indicates that the three components of the vectors in the population have the same variance. Also, elements x_1 and x_2, as well as x_1 and x_3, are positively correlated; elements x_2 and x_3 are negatively correlated. ■

Because $\mathbf{C_x}$ is real and symmetric, finding a set of n orthonormal eigenvectors always is possible (Noble and Daniel [1988]). Let $\mathbf{e}_i$ and λ_i, $i = 1, 2, \ldots, n$, be the eigenvectors and corresponding eigenvalues of $\mathbf{C_x}$,† arranged (for convenience) in descending order so that $\lambda_j \geq \lambda_{j+1}$ for $j = 1, 2, \ldots, n-1$. Let $\mathbf{A}$ be a matrix whose rows are formed from the eigenvectors of $\mathbf{C_x}$, ordered so that the first row of $\mathbf{A}$ is the eigenvector corresponding to the largest eigenvalue, and the last row is the eigenvector corresponding to the smallest eigenvalue.

Suppose that we use $\mathbf{A}$ as a transformation matrix to map the $\mathbf{x}$'s into vectors denoted by $\mathbf{y}$'s, as follows:

$$\mathbf{y} = \mathbf{A}(\mathbf{x} - \mathbf{m_x}). \tag{11.4-6}$$

This expression is called the *Hotelling transform*, which, as will be shown shortly, has some interesting and useful properties.

It is not difficult to show that the mean of the $\mathbf{y}$ vectors resulting from this transformation is zero; that is,

$$\mathbf{m_y} = E\{\mathbf{y}\} = \mathbf{0}. \tag{11.4-7}$$

It follows from basic matrix theory that the covariance matrix of the $\mathbf{y}$'s is given in terms of $\mathbf{A}$ and $\mathbf{C_x}$ by the expression

$$\mathbf{C_y} = \mathbf{A}\mathbf{C_x}\mathbf{A}^T. \tag{11.4-8}$$

Furthermore, because of the way $\mathbf{A}$ was formed, $\mathbf{C_y}$ is a diagonal matrix whose elements along the main diagonal are the eigenvalues of $\mathbf{C_x}$; that is,

$$\mathbf{C_y} = \begin{bmatrix} \lambda_1 & & & 0 \\ & \lambda_2 & & \\ & & \ddots & \\ 0 & & & \lambda_n \end{bmatrix}. \tag{11.4-9}$$

The off-diagonal elements of this covariance matrix are 0, so the elements of the $\mathbf{y}$ vectors are uncorrelated. Keep in mind that the λ_j's are the eigenvalues of $\mathbf{C_x}$ and that the elements along the main diagonal of a diagonal matrix are its eigenvalues (Noble and Daniel [1988]). Thus $\mathbf{C_x}$ and $\mathbf{C_y}$ have the same eigenvalues. In fact, the same is true for the eigenvectors.

† By definition, the eigenvectors and eigenvalues of an $n \times n$ matrix, $\mathbf{C}$, satisfy the relation $\mathbf{C}\mathbf{e}_i = \lambda_i \mathbf{e}_i$, for $i = 1, 2, \ldots, n$.

Another important property of the Hotelling transform deals with the reconstruction of $\mathbf{x}$ from $\mathbf{y}$. Because the rows of $\mathbf{A}$ are orthonormal vectors, it follows that $\mathbf{A}^{-1} = \mathbf{A}^T$, and any vector $\mathbf{x}$ can be recovered from its corresponding $\mathbf{y}$ by using the expression

$$\mathbf{x} = \mathbf{A}^T\mathbf{y} + \mathbf{m}_\mathbf{x}. \tag{11.4-10}$$

Suppose, however, that instead of using all the eigenvectors of $\mathbf{C}_\mathbf{x}$ we form matrix $\mathbf{A}_k$ from the k eigenvectors corresponding to the k largest eigenvalues, yielding a transformation matrix of order $k \times n$. The $\mathbf{y}$ vectors would then be k dimensional, and the reconstruction given in Eq. (11.4-10) would no longer be exact (this is somewhat analogous to the procedure we used in Section 11.2.3 to describe a boundary with a few Fourier coefficients).

The vector reconstructed by using $\mathbf{A}_k$ is

$$\hat{\mathbf{x}} = \mathbf{A}_k^T\mathbf{y} + \mathbf{m}_\mathbf{x}. \tag{11.4-11}$$

It can be shown that the mean square error between $\mathbf{x}$ and $\hat{\mathbf{x}}$ is given by the expression

$$\begin{aligned} e_{\text{ms}} &= \sum_{j=1}^{n} \lambda_j - \sum_{j=1}^{k} \lambda_j \\ &= \sum_{j=k+1}^{n} \lambda_j. \end{aligned} \tag{11.4-12}$$

The first line of Eq. (11.4-12) indicates that the error is zero if $k = n$ (that is, if all the eigenvectors are used in the transformation). Because the λ_j's decrease monotonically, Eq. (11.4-12) also shows that the error can be minimized by selecting the k eigenvectors associated with the largest eigenvalues. Thus the Hotelling transform is optimal in the sense that it minimizes the mean square error between the vectors $\mathbf{x}$ and their approximations $\hat{\mathbf{x}}$. Due to this idea of using the eigenvectors corresponding to the largest eigenvalues, the Hotelling transform also is known as the *principal components* transform.

EXAMPLE 11.10: Use of principal components to describe images.

■ Figure 11.26 shows six images generated by a 6-band multispectral scanner operating in the wavelengths shown in Table 11.4. Viewing the images as shown in Fig. 11.27 allows formation of a 6-dimensional vector $\mathbf{x} = (x_1, x_2, \ldots, x_6)^T$ from each set of corresponding pixels in the images, as discussed at the beginning of this section. The images in this particular application are of resolution 384×239 so the population consists of 91,776 vectors from which to compute the mean vector and covariance matrix. Table 11.5 shows the eigenvalues of $\mathbf{C}_\mathbf{x}$. Note the dominance of the first two eigenvalues.

Use of Equation (11.4-6) generated a set of transformed $\mathbf{y}$ vectors corresponding to the $\mathbf{x}$ vectors. From them, six principal component images were assembled (images are constructed from vectors simply by applying Fig. 11.27 in reverse). Figure 11.28 shows the results. Component 1 denotes the image formed from all the y_1 components of the transformed vectors, and so on for the other five images. Recall from basic matrix theory that y_1, for example, is obtained by performing the inner (dot) product of the first row of $\mathbf{A}$ with the column vector $(\mathbf{x} - \mathbf{m}_\mathbf{x})^T$.

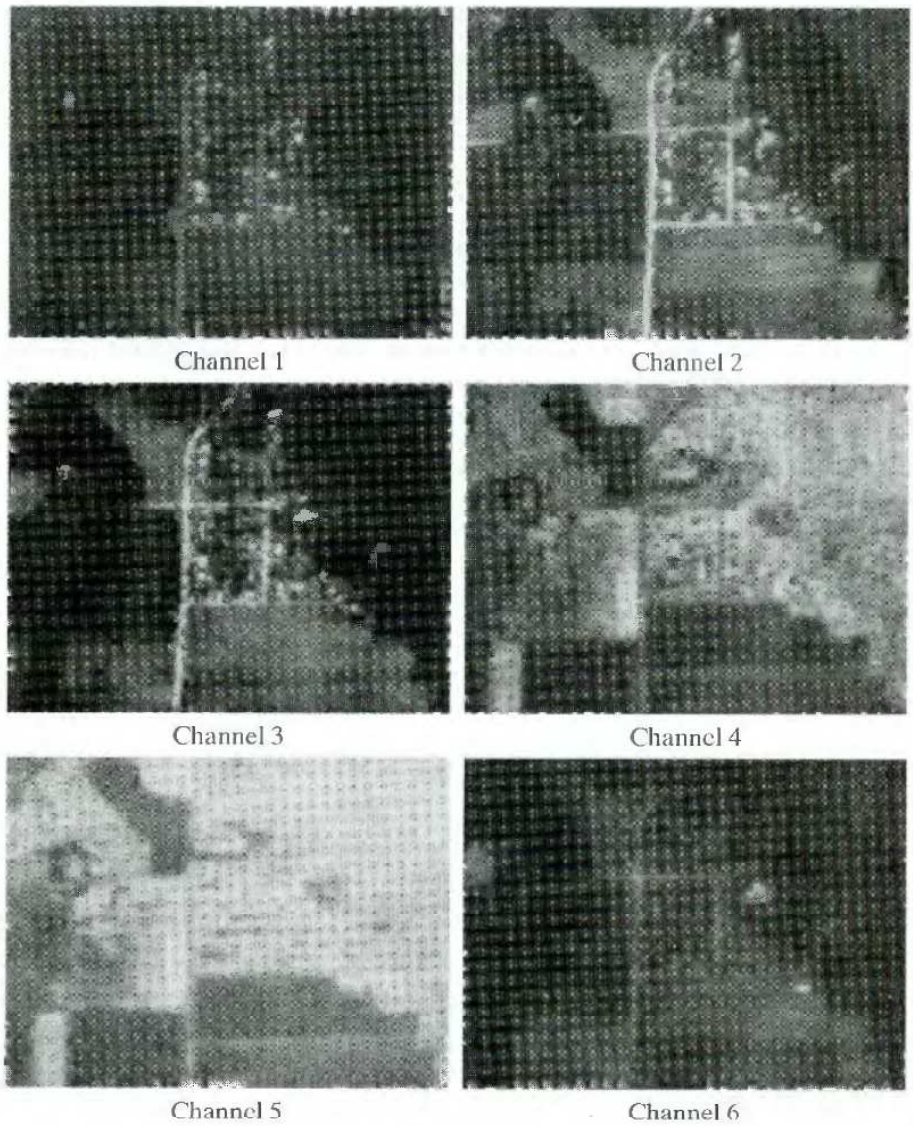


FIGURE 11.26 Six spectral images from an airborne scanner. (Courtesy of the Laboratory for Applications of Remote Sensing, Purdue University.)

TABLE 11.4 Channel numbers and wavelengths.

Channel	Wavelength band (microns)
1	0.40–0.44
2	0.62–0.66
3	0.66–0.72
4	0.80–1.00
5	1.00–1.40
6	2.00–2.60

The first row of **A** is the eigenvector corresponding to the largest eigenvalue of the covariance matrix of the population, and this eigenvalue gives the variance of the gray levels of the first transformed image. Thus based on the numbers shown in Table 11.5, this image should have the highest contrast. That such is the case is quite evident in Fig. 11.28. Because the first two images account for about

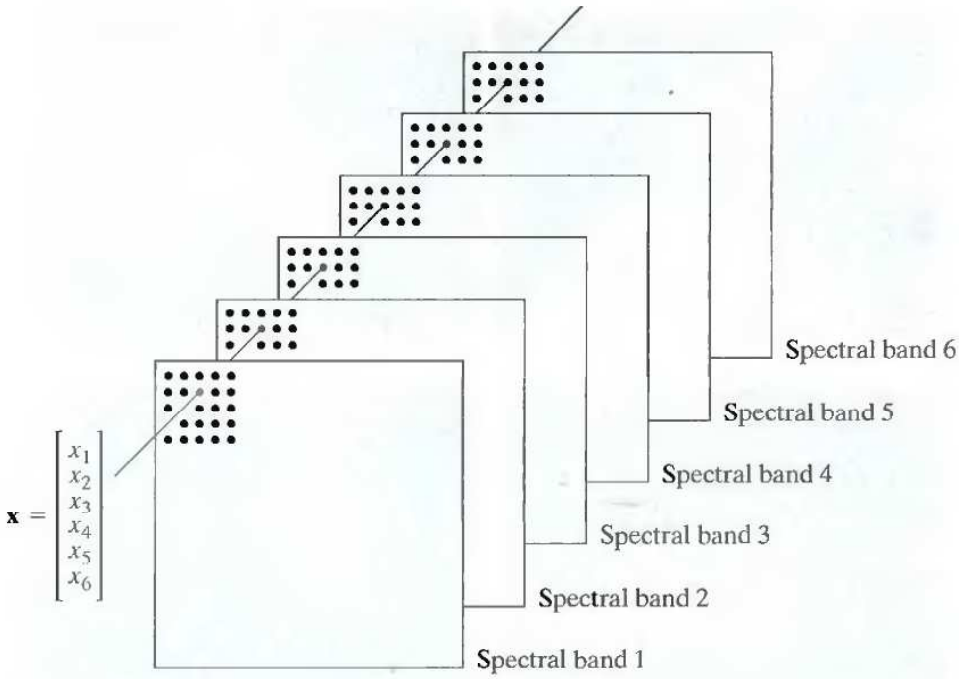


FIGURE 11.27 Formation of a vector from corresponding pixels in six images.

λ_1	λ_2	λ_3	λ_4	λ_5	λ_6
3210	931.4	118.5	83.88	64.00	13.40

TABLE 11.5 Eigenvalues of the covariance matrix obtained from the images in Fig. 11.26.

94% of the total variance, the fact that the other four principal-component images have low contrast is not unexpected. Thus if instead of storing all six images for posterity, only the first two transformed images, along with $\mathbf{m}_x$ and the first two rows of $\mathbf{A}$, were stored, a credible job of reconstructing an approximation to the six original images could be done at a later date. This capability for performing *data compression*, although not impressive by today's standards is a useful byproduct of the Hotelling transform. In terms of description, this means describing the content of six images with two, plus the mean vector and first two rows of the transformation matrix. The same argument would apply if instead of entire images we were discussing regions. ■

EXAMPLE 11.11: Use of principal components for describing boundaries and regions in a single image.

■ In the preceding discussion we showed how to apply the principal components transformation to sets of images or regions. In this example we illustrate how to use principal components for describing boundaries and regions in a single image. The approach is to form two-dimensional vectors from the *coordinates* of the boundary or region. Consider the object shown in Fig. 11.29(a). Vectors are formed from the coordinates of the pixels in the object if we wish

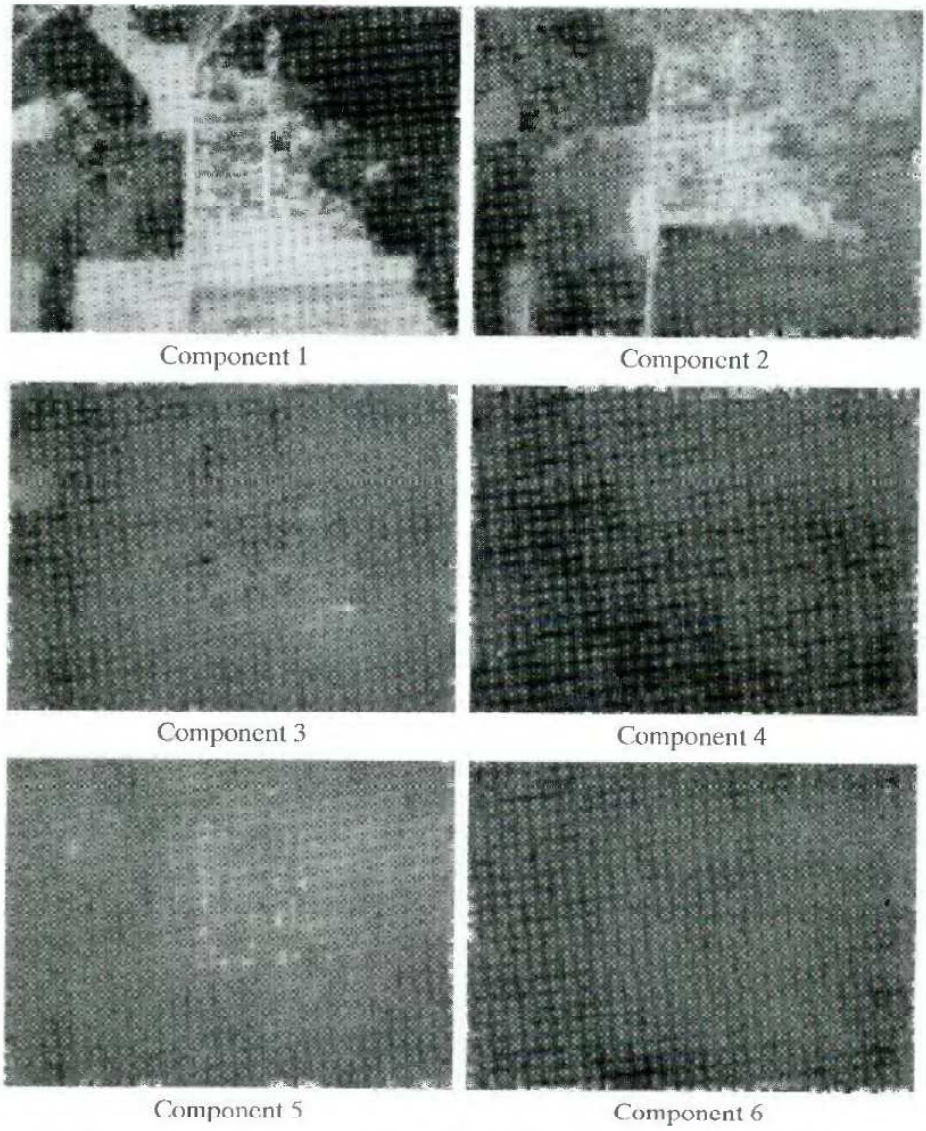


FIGURE 11.28 Six principal-component images computed from the data in Fig. 11.26. (Courtesy of the Laboratory for Applications of Remote Sensing, Purdue University.)

to describe the region. If we wish to describe the boundary, we only use the coordinates of the points on the boundary.

The resulting vectors then are treated as a 2-D population of random vectors. In other words, each pixel in the object is treated as a 2-D vector $\mathbf{x} = (a, b)^T$, where a and b are the coordinate values of that pixel with respect to the x_1- and x_2-axes. These vectors are used to compute the mean vector and covariance matrix of the population (object). The problem is much simpler than before because we are working in only two dimensions.

The net effect of using Eq. (11.4-6) is to establish a new coordinate system whose origin is at the centroid of the population (the coordinates of the mean vector) and whose axes are in the direction of the eigenvectors of $\mathbf{C_x}$, as shown in Fig. 11.29(b). This coordinate system clearly shows that the transformation in Eq. (11.4-6) is a rotation transformation that aligns the data with the eigenvectors,