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

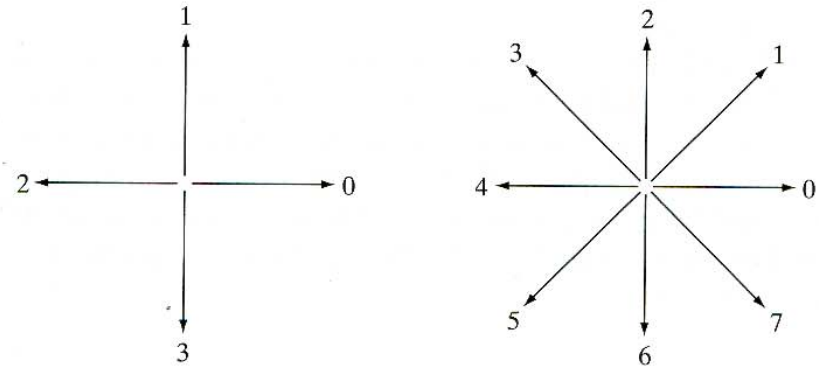
Chain codes are used to represent a boundary by a connected sequence of straight-line segments of specified length and direction. Typically, this representation is based on 4- or 8-connectivity of the segments. The direction of each segment is coded by using a numbering scheme such as the ones shown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

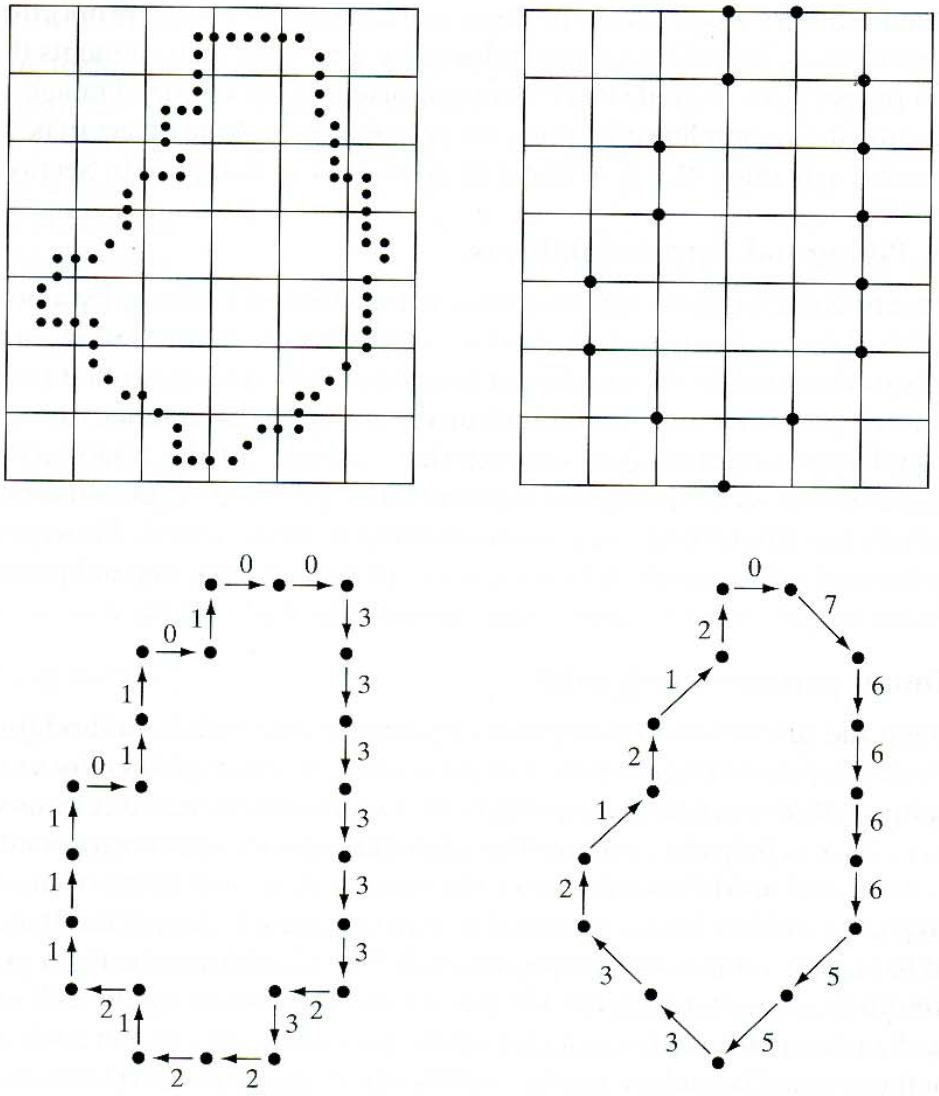
Digital images usually are acquired and processed in a grid format with equal spacing in the x- and y-directions, so a chain code could be generated by following a boundary in, say, a clockwise direction and assigning a direction to the segments connecting every pair of pixels. This method generally 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

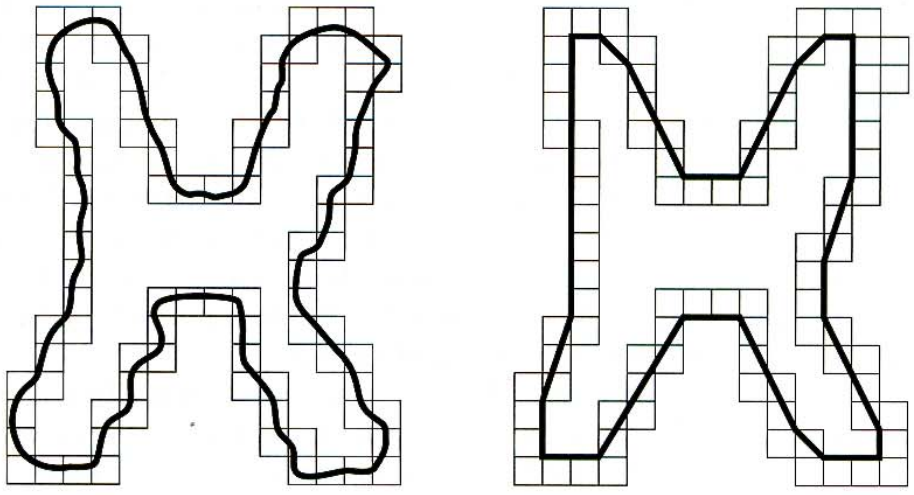
different boundary shapes, with the degree of dissimilarity being proportional to image resolution. This effect can be reduced by selecting chain elements that are large in proportion to the distance between pixels in the digitized image and/or by orienting the resampling grid along the principal axes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

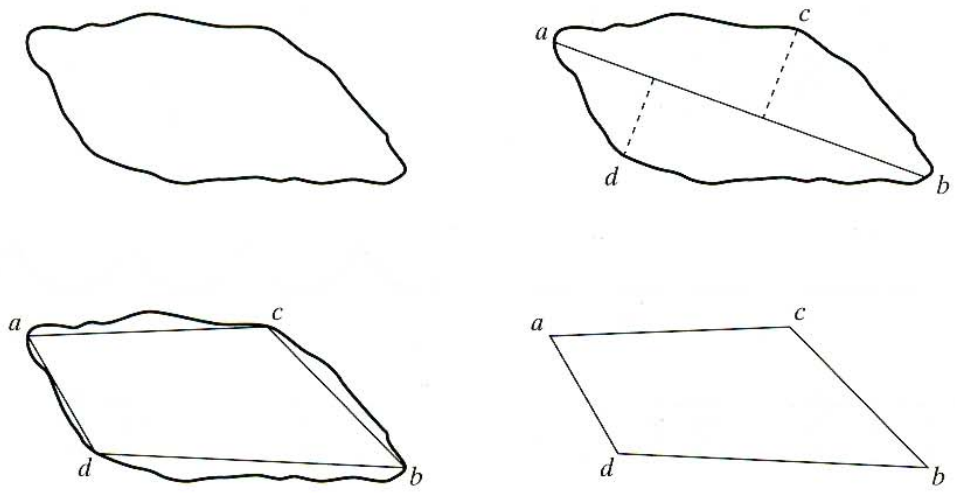
FIGURE 11.3 (a) Object boundary enclosed by cells. (b) Minimum perimeter polygon.

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

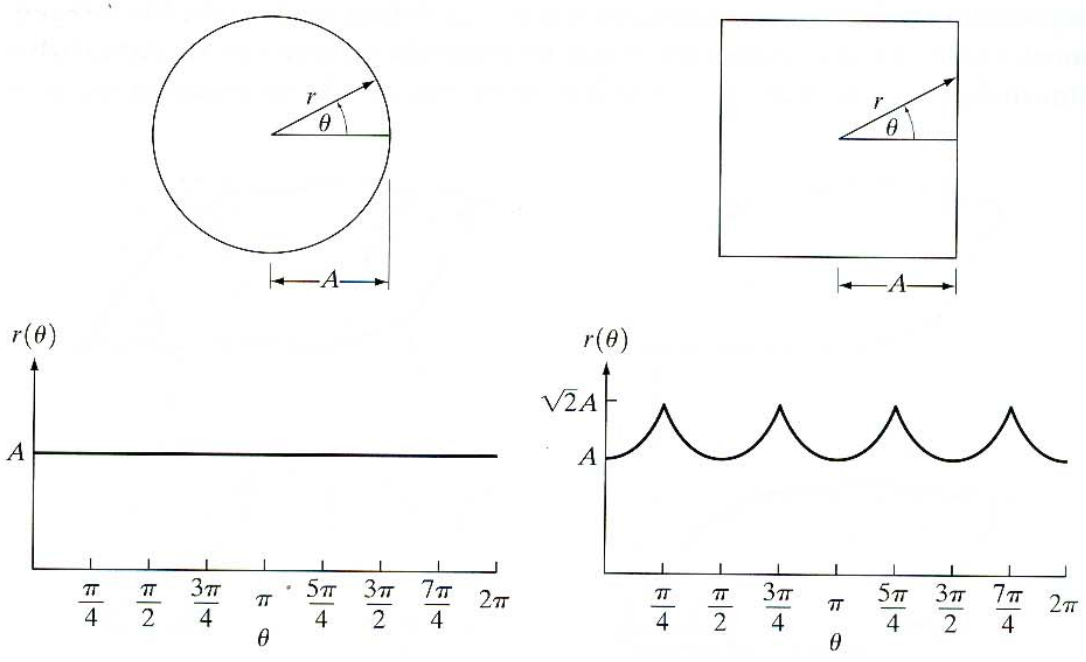


FIGURE 11.5 Distance-versus-angle signatures. In (a) $r(\theta)$ is constant. In (b), the signature consists of repetitions of the pattern $r(\theta) = A \sec\theta$ for $0 \le \theta \le \pi/4$ and $r(\theta) = A \csc\theta$ for $\pi/4 < \theta \le \pi/2$.

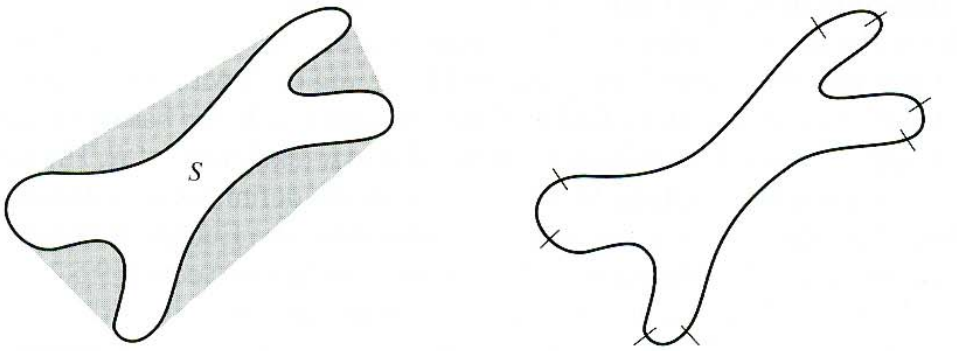
same range of values, say, [0, 1]. The main advantage of this method is simplicity, but it has the potentially serious disadvantage that scaling of the entire function depends on only two values: the minimum and maximum. If the shapes are noisy, this dependence can be a source of error from object to 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.

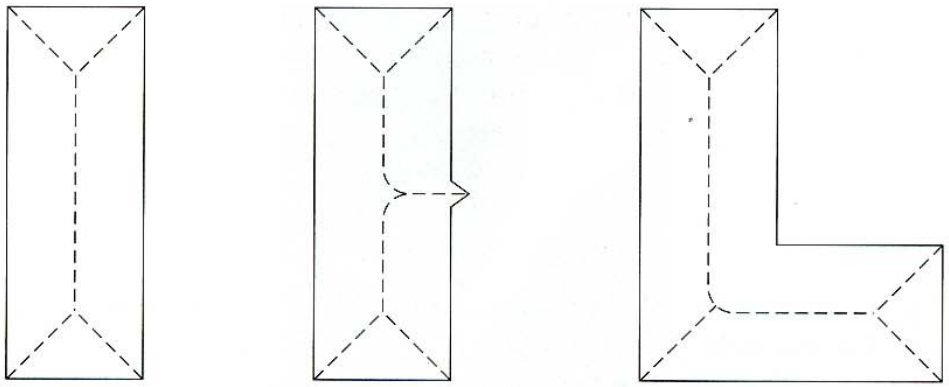
The concepts of a convex hull and its deficiency are equally useful for describing an entire region, as well as just its boundary. For example, description of a region might be based on its area and the area of its convex deficiency, the number of components in the convex deficiency, the relative location 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.

$$\begin{matrix} 0 & 0 & 1 \\ 1 & p_1 & 0 \\ 1 & 0 & 1 \end{matrix}$$

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

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 $p_8 = 0$.

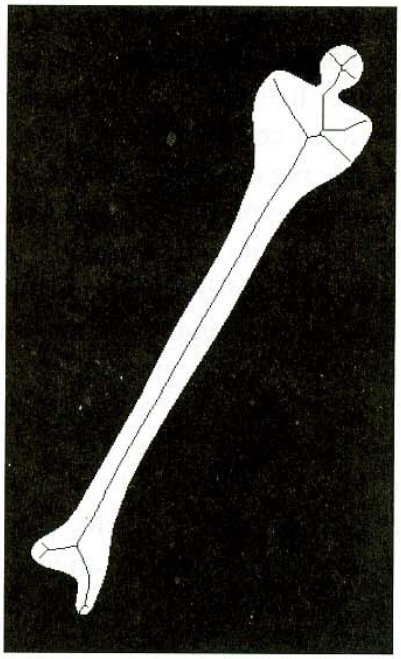


FIGURE 11.10 Human leg bone and skeleton of the region shown superimposed.

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.

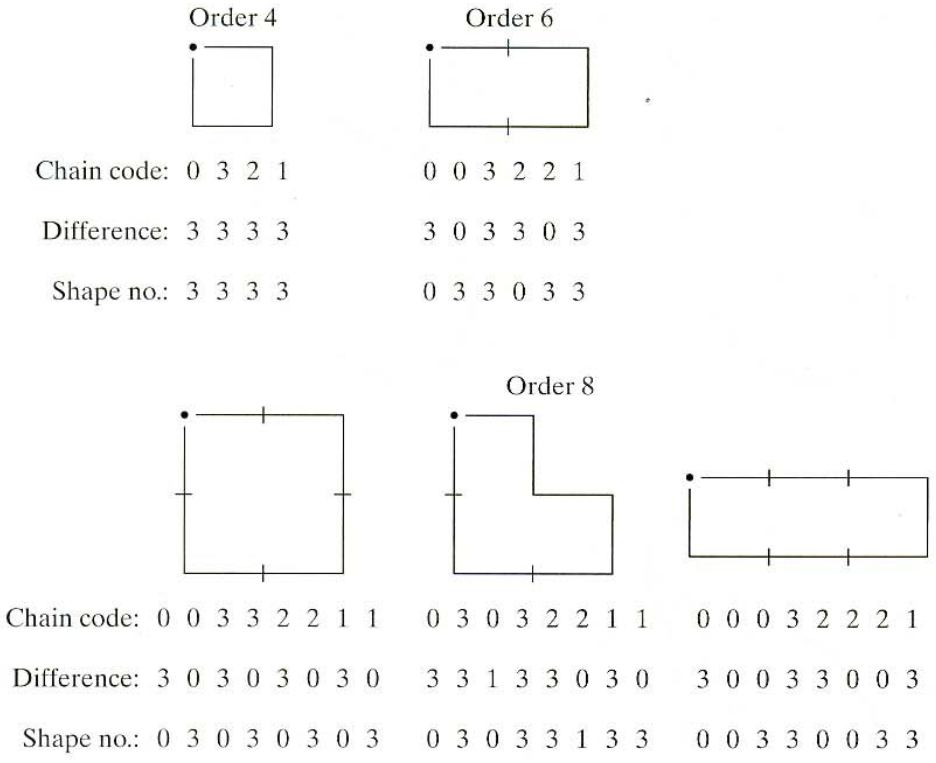


FIGURE 11.11 All shapes of order 4, 6, and 8. The directions are from Fig. 11.1(a), and the dot indicates the starting point.

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). ■

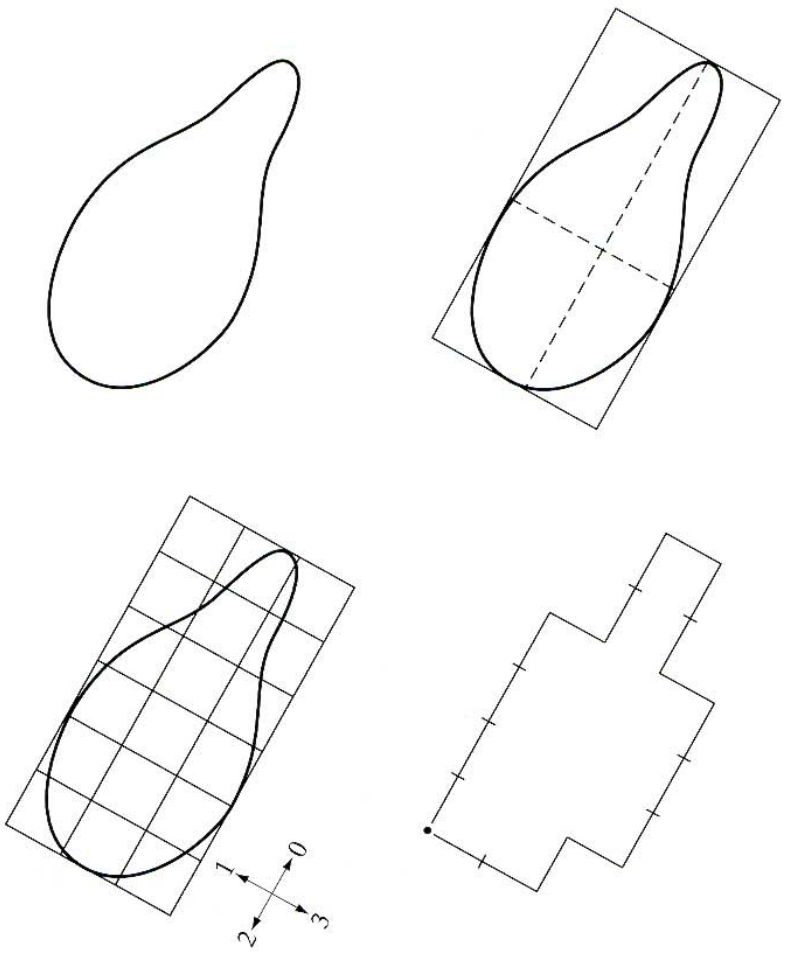
11.2.3 Fourier Descriptors

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,

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

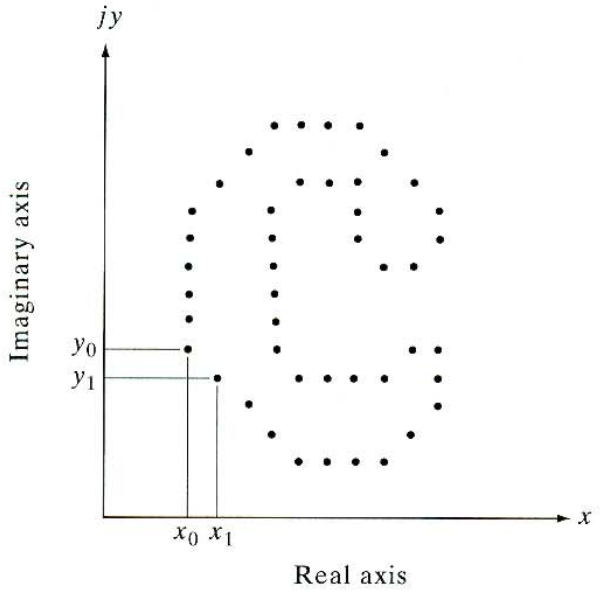


FIGURE 11.13 A digital boundary and its representation as a complex sequence. The points (x_0, y_0) and (x_1, y_1) shown are (arbitrarily) the first two points in the sequence.

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. ■

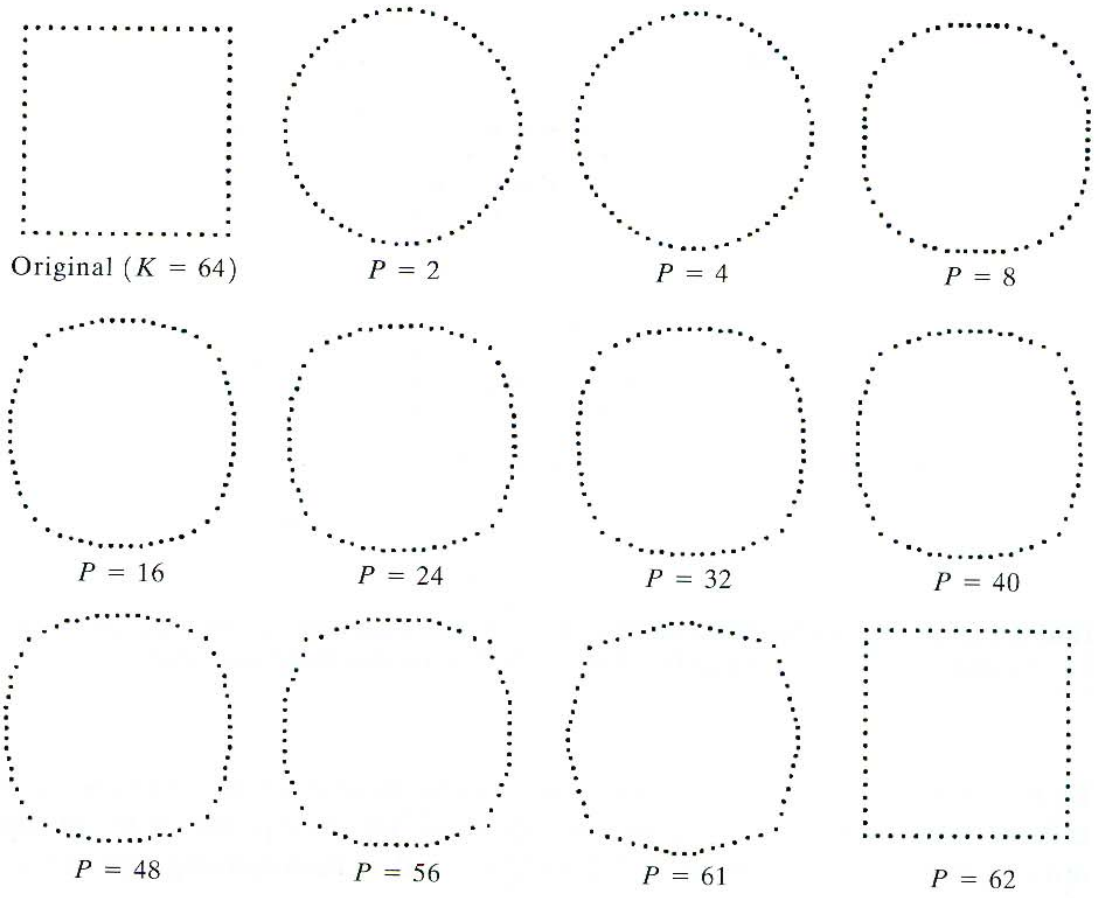


FIGURE 11.14 Examples of reconstruction from Fourier descriptors. P is the number of Fourier coefficients used in the reconstruction of the boundary.

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 $e^{j\theta}$.

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 Some basic properties of Fourier descriptors.

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.

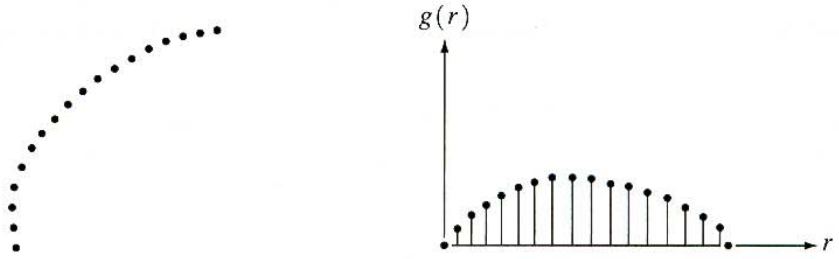
Let us treat the amplitude of g as a discrete random variable v and form an amplitude histogram $p(v_i), i = 0, 1, 2, \ldots, A - 1$, where A is the number of discrete amplitude increments in which we divide the amplitude scale. Then, keeping in mind that $p(v_i)$ is an estimate of the probability of 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. ■

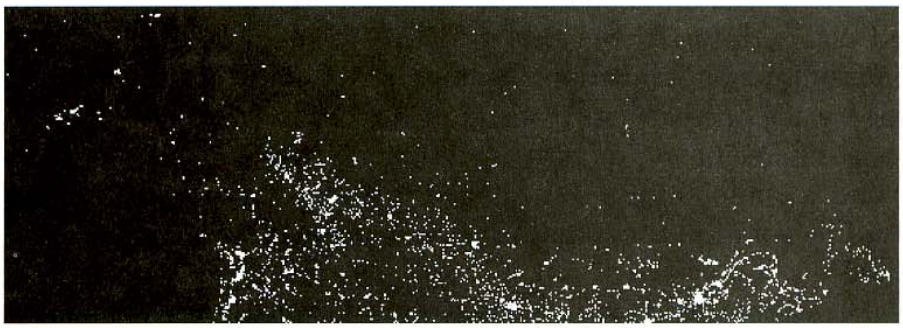
11.3.2 Topological Descriptors

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.)

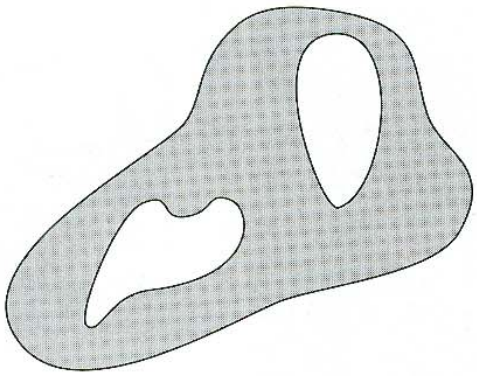


FIGURE 11.17 A region with two holes.

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:

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

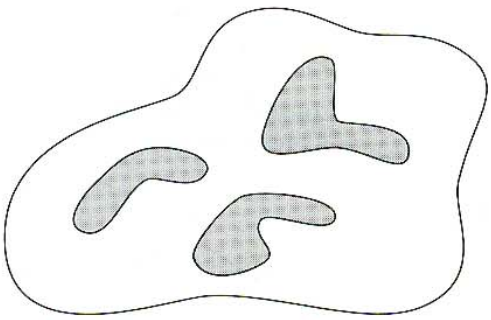
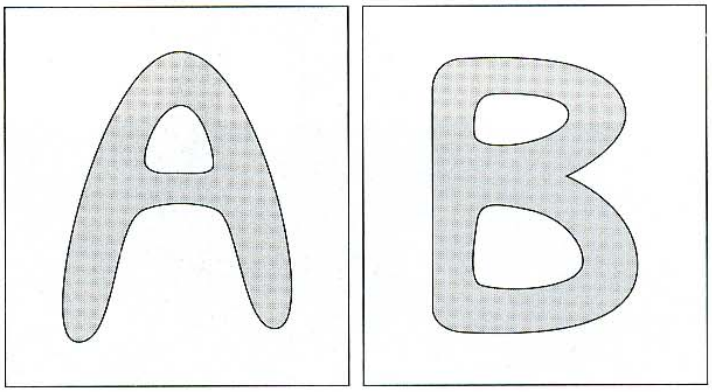


FIGURE 11.18 A region with three connected components.



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.

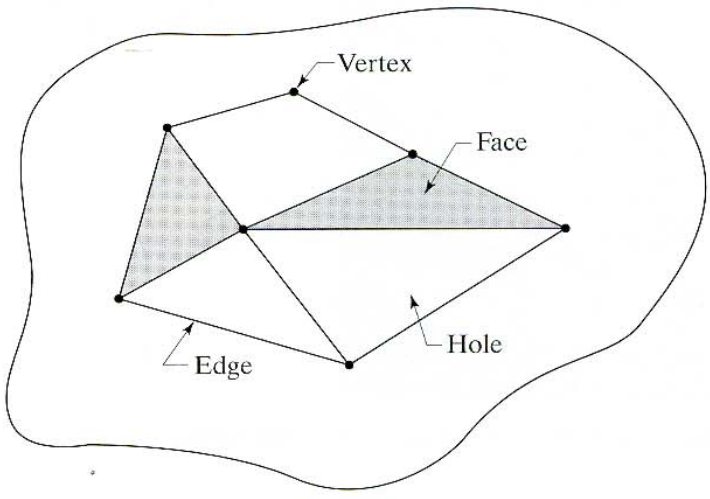
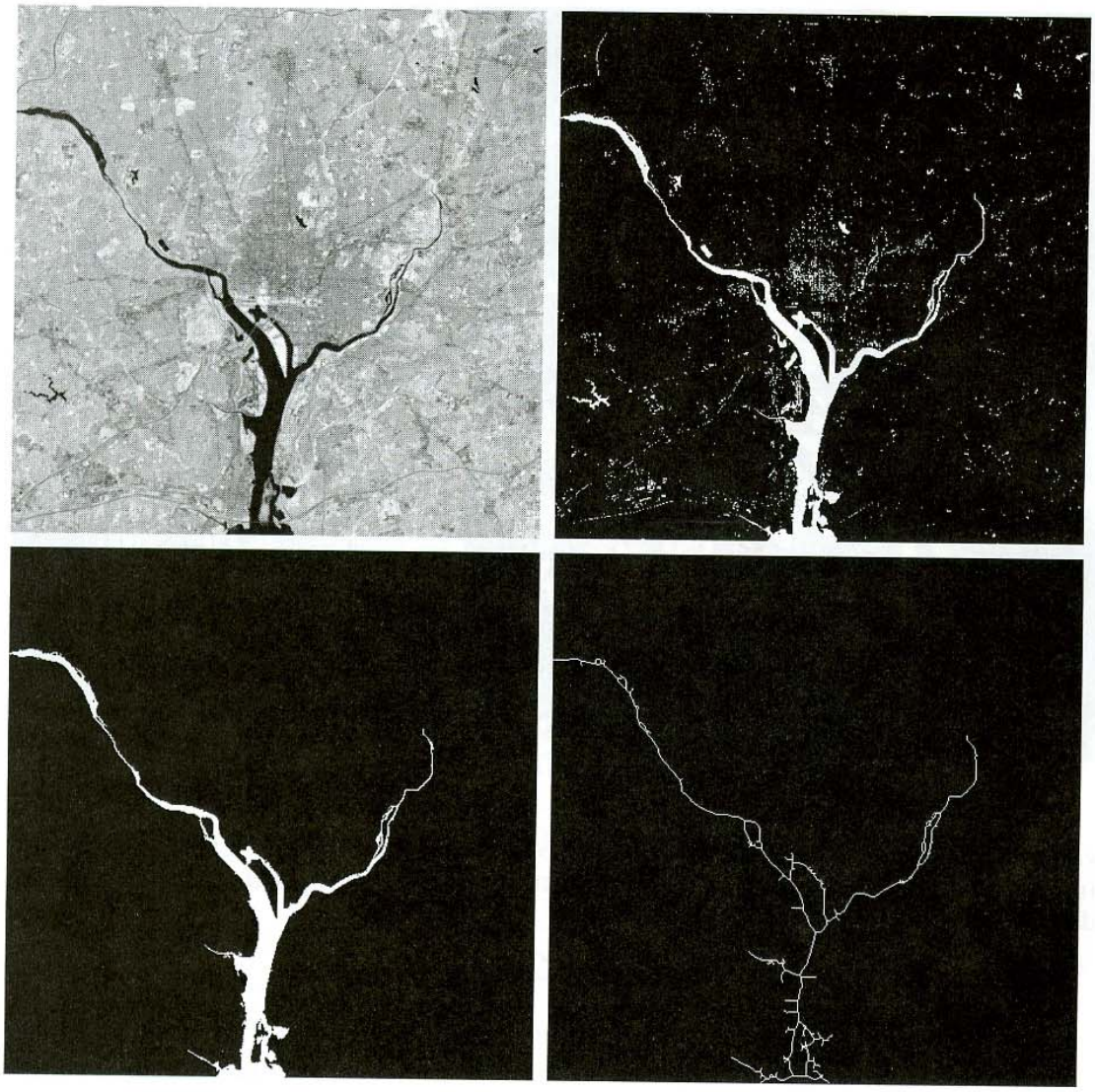


FIGURE 11.20 A region containing a polygonal network.



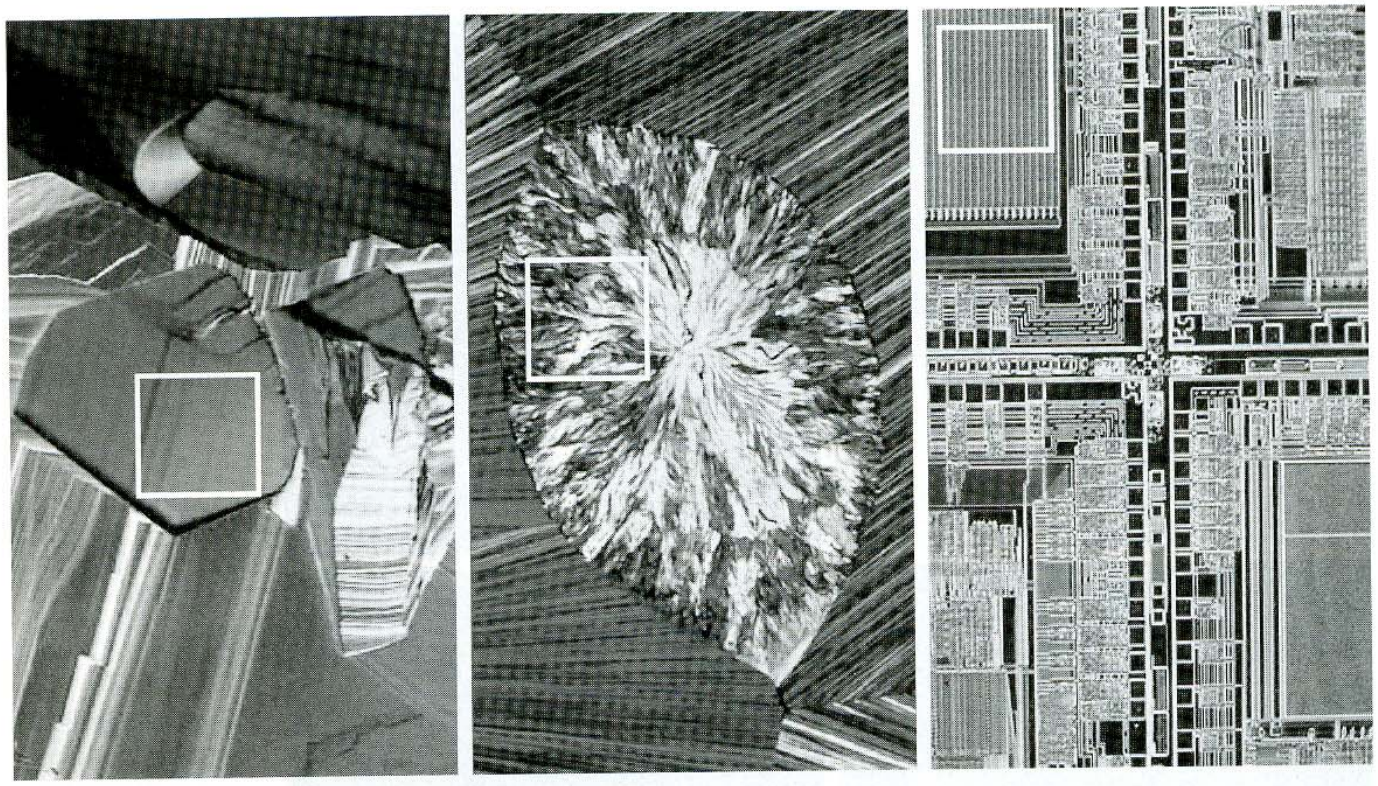
a b
c d

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

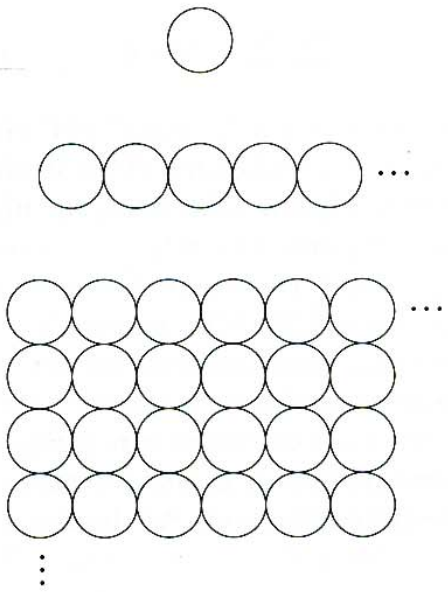


FIGURE 11.23
(a) Texture primitive.
(b) Pattern generated by the rule $S \rightarrow aS$.
(c) 2-D texture pattern generated by this and other rules.

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

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

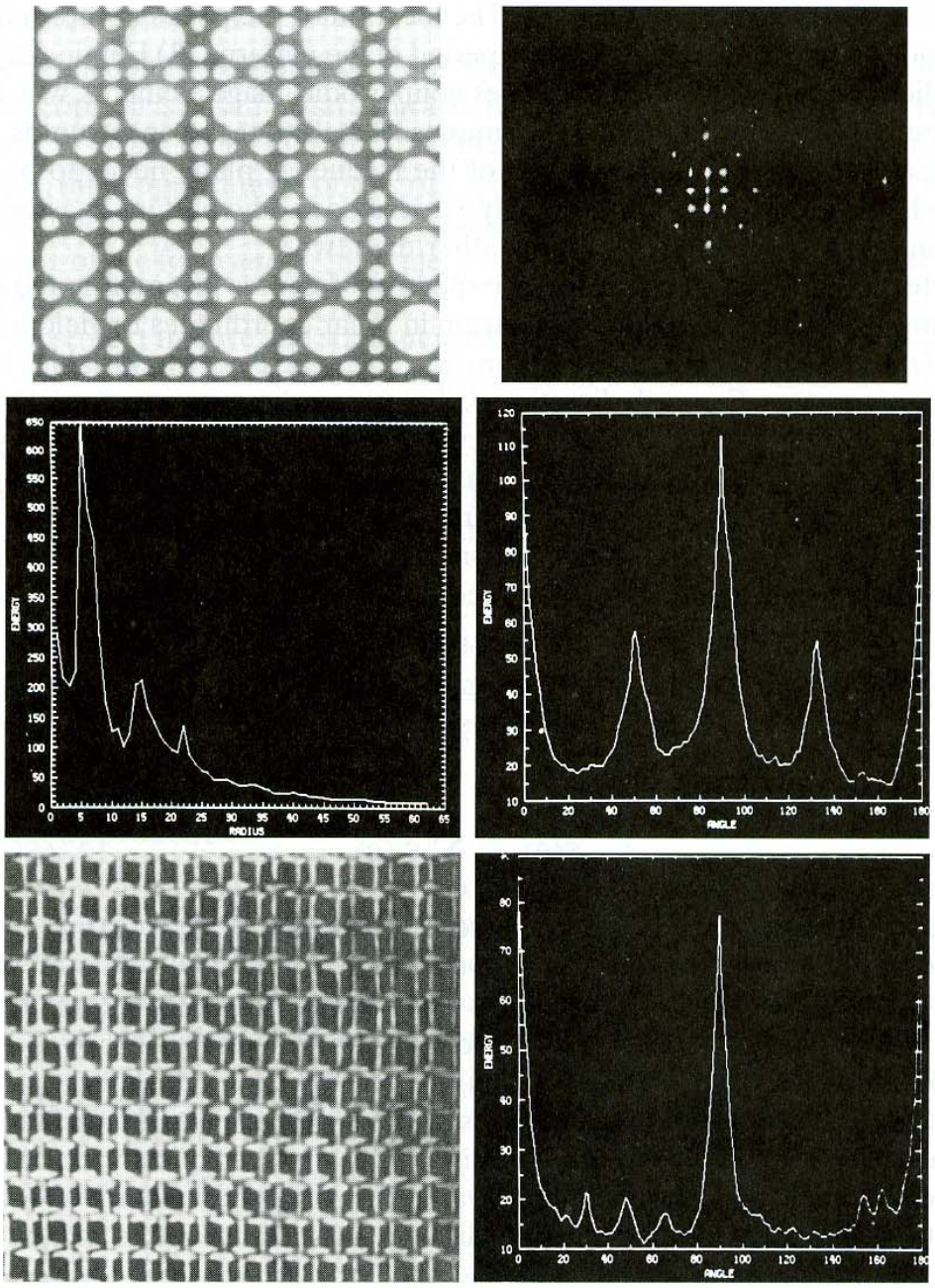
where R_0 is the radius of a circle centered at the origin.

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]).

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

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

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

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

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

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

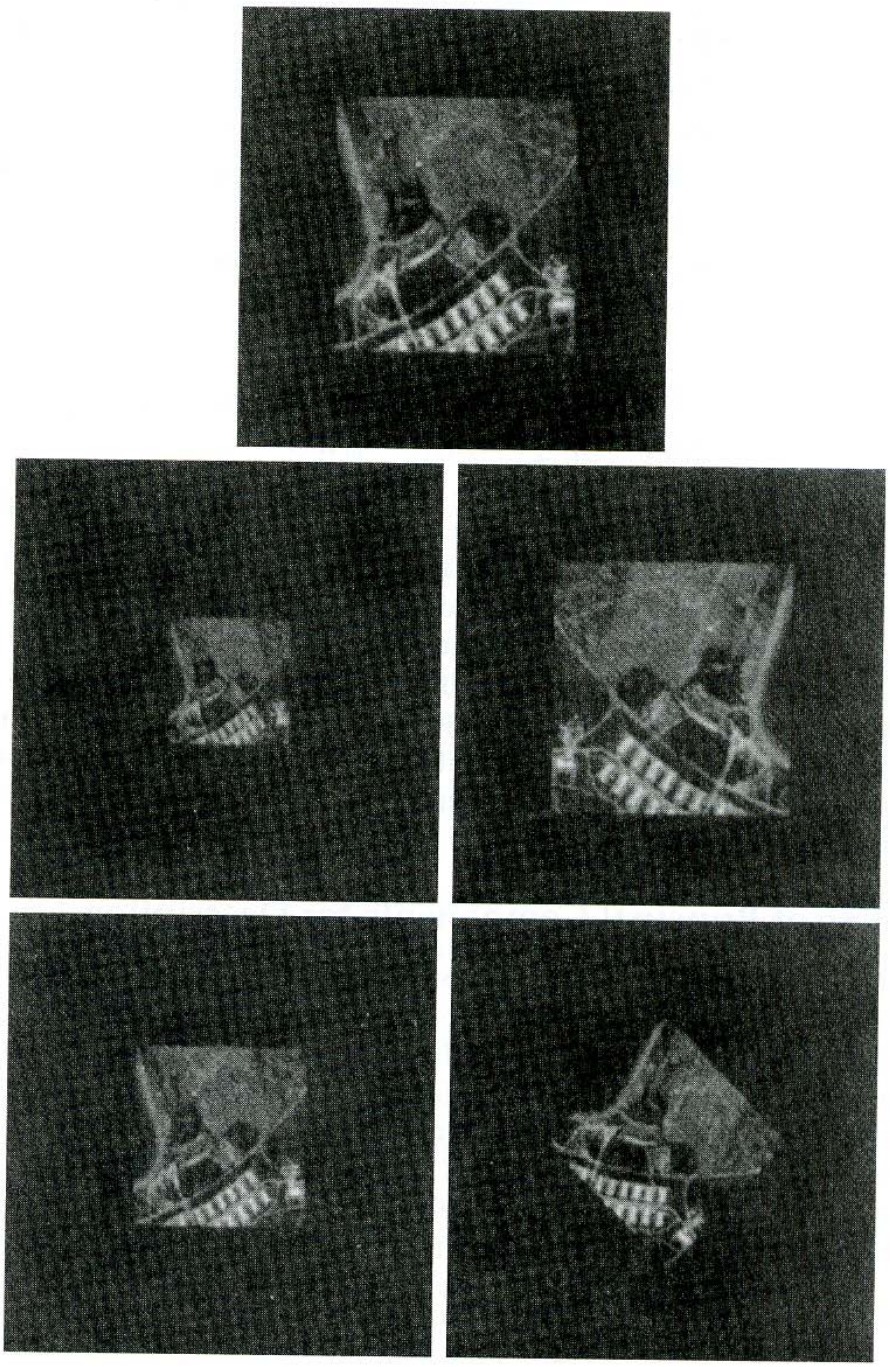


FIGURE 11.25
Images used to demonstrate properties of moment invariants (see Table 11.3).

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:

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

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 $\mathbf{C}_\mathbf{x}$ is real and symmetric, finding a set of n orthonormal eigenvectors always is possible (Noble and Daniel [1988]). Let $\mathbf{e}_i$ and $\lambda_i, i = 1, 2, \ldots, n$, be the eigenvectors and corresponding eigenvalues of $\mathbf{C}_\mathbf{x}$,[†] 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 $\mathbf{C}_\mathbf{x}$, 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:

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

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,

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

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

$$\mathbf{C}_\mathbf{y} = \mathbf{A}\mathbf{C}_\mathbf{x}\mathbf{A}^T. \tag{11.4-8}$$

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

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

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 $\mathbf{C}_\mathbf{x}$ and that the elements along the main diagonal of a diagonal matrix are its eigenvalues (Noble and Daniel [1988]). Thus $\mathbf{C}_\mathbf{x}$ and $\mathbf{C}_\mathbf{y}$ 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 $(\mathbf{x} - \mathbf{m}_\mathbf{x})^T$.

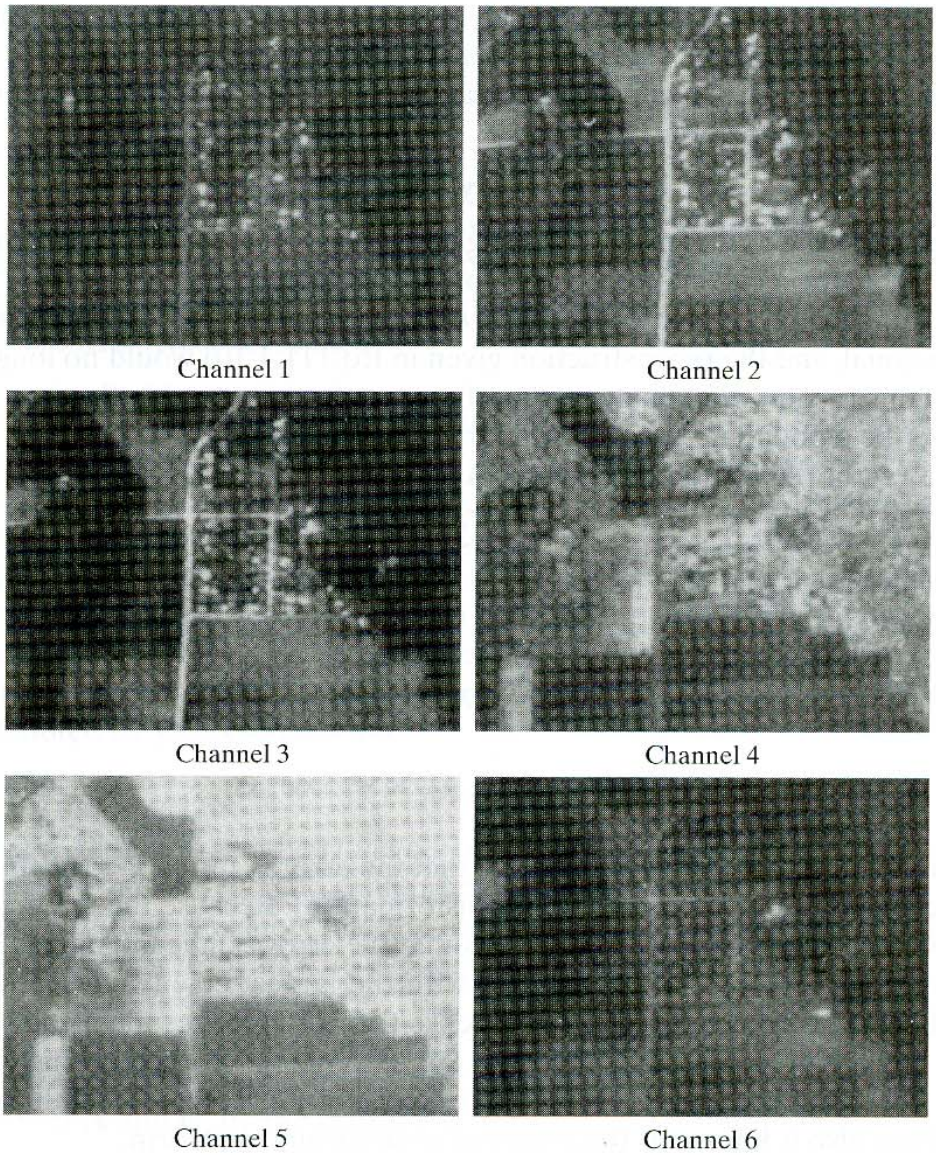


FIGURE 11.26 Six spectral images from an airborne scanner. (Courtesy of the Laboratory for Applications of Remote Sensing, Purdue University.)

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

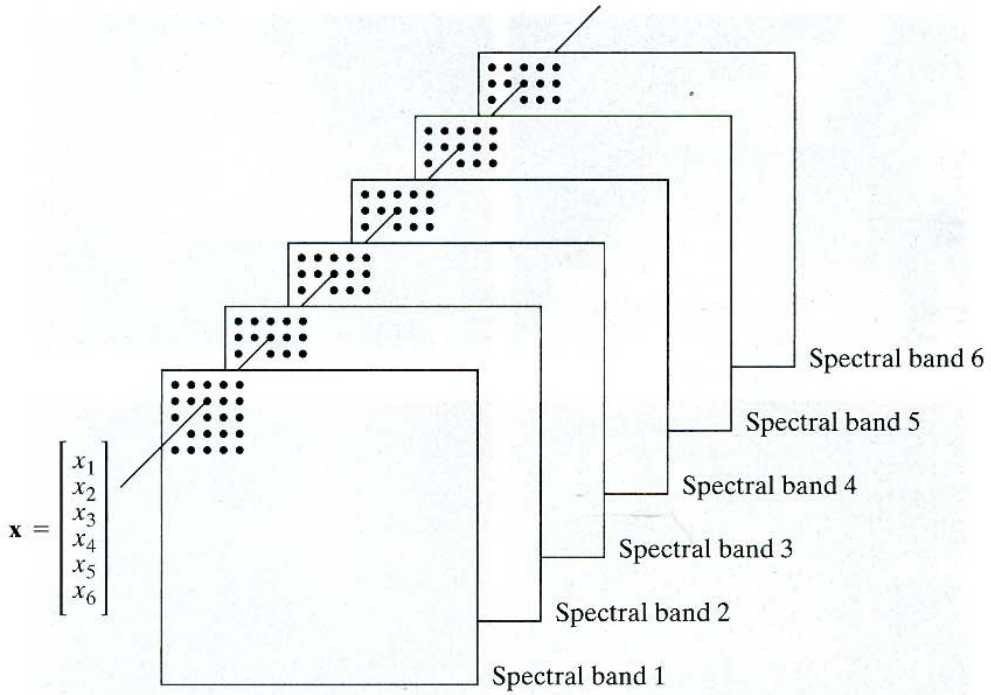


FIGURE 11.27 Formation of a vector from corresponding pixels in six images.

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. ■

■ 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

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

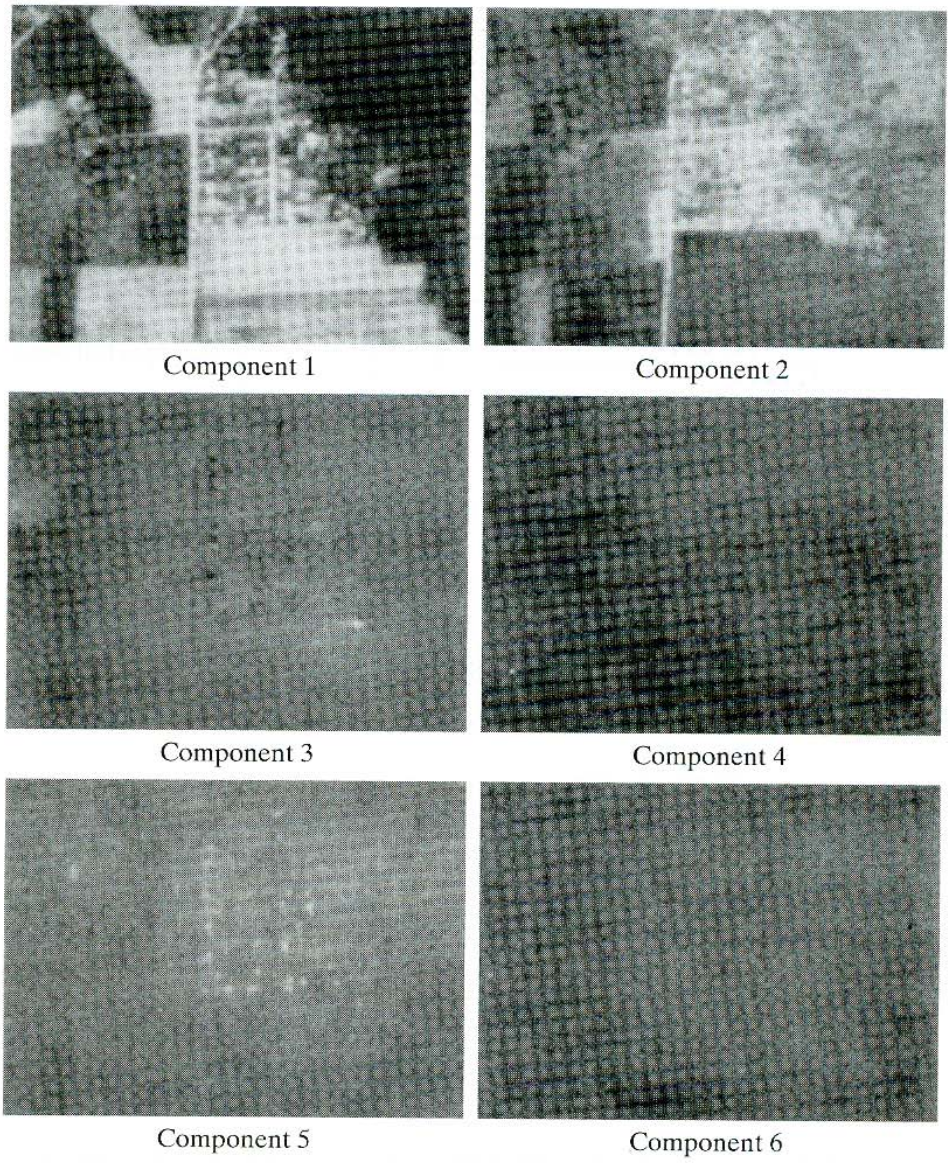
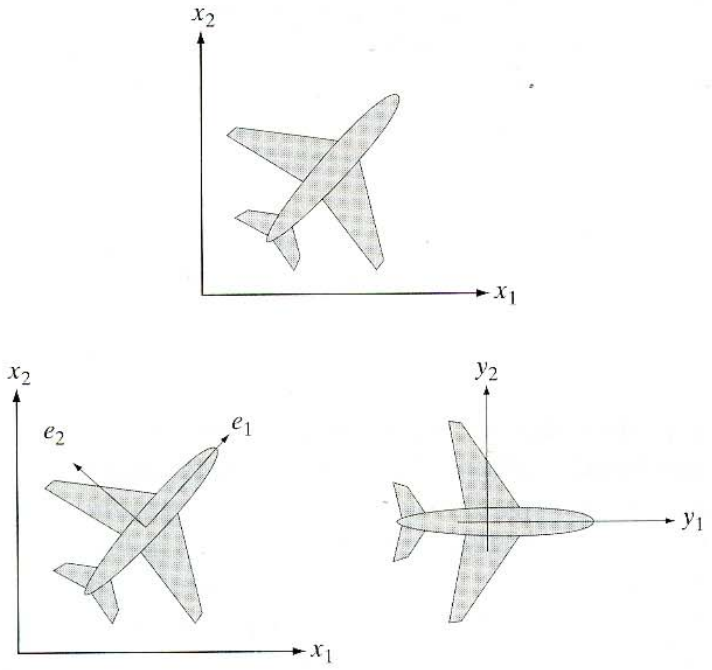


FIGURE 11.28 Six principal-component images computed from the data in Fig. 11.26. (Courtesy of the Laboratory for Applications of Remote Sensing, Purdue University.)

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 $\mathbf{C}_\mathbf{x}$, 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,



a
b c

FIGURE 11.29 (a) An object. (b) Eigenvectors. (c) Object rotated by using Eq. (11.4-6). The net effect is to align the object along its eigen axes.

as shown in Fig. 11.29(c). In fact, this alignment is precisely the mechanism that decorrelates the data. Furthermore, as the eigenvalues appear along the main diagonal of $\mathbf{C}_y$, λ_i is the variance of component y_i along eigenvector $\mathbf{e}_i$. The two eigenvectors are perpendicular. The y-axes sometimes are called the *eigen axes*, for obvious reasons. ■

The concept of aligning a 2-D object with its principal eigenvectors plays an important role in description. As noted earlier, description should be as independent as possible to variations in size, translation, and rotation. The ability to *align* the object with its principal axes provides a reliable means for removing the effects of rotation. The eigenvalues are the variances along the eigen axes, and can be used for size normalization. The effects of translation are accounted for by centering the object about its mean, as shown in Eq. (11.4-6). Keep in mind the fact that the method of description derived in this section is equally applicable to both regions and boundaries.

11.5 Relational Descriptors

We introduced in Section 11.3.3 the concept of rewriting rules for describing texture. In this section we expand that concept in the context of relational descriptors. These apply equally well to boundaries or regions, and their main purpose is to capture in the form of rewriting rules basic repetitive patterns in a boundary or region.

a b

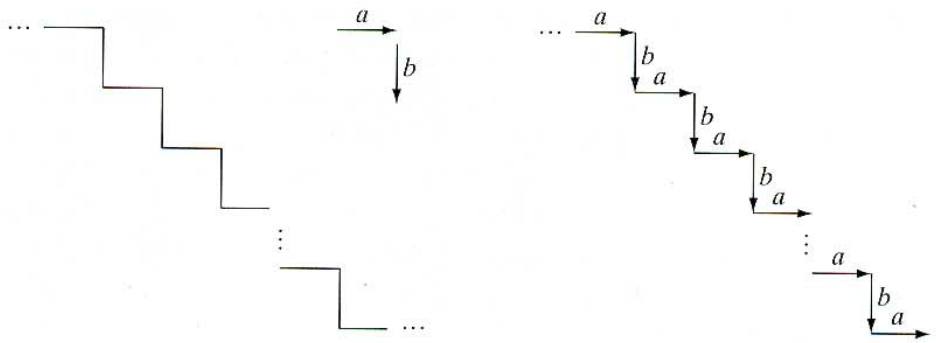


FIGURE 11.30
(a) A simple staircase structure. (b) Coded structure.

Consider the simple staircase structure shown in Fig. 11.30(a). Assume that this structure has been segmented out of an image and that we want to describe it in some formal way. By defining the two *primitive elements* a and b shown, we may code Fig. 11.30(a) in the form shown in Fig. 11.30(b). The most obvious property of the coded structure is the repetitiveness of the elements a and b. Therefore, a simple description approach is to formulate a recursive relationship involving these primitive elements. One possibility is to use the *rewriting rules*:

(a) $S \rightarrow aA$,
(b) $A \rightarrow bS$, and
(c) $A \rightarrow b$,

where S and A are variables and the elements a and b are constants corresponding to the primitives just defined. Rule 1 indicates that S, called the *starting symbol*, can be replaced by primitive a and variable A. This variable, in turn, can be replaced by b and S or by b alone. Replacing A with bS, leads back to the first rule and the procedure can be repeated. Replacing A with b terminates the procedure, because no variables remain in the expression. Figure 11.31 illustrates some sample derivations of these rules, where the numbers below the structures represent the order in which rules 1, 2, and 3 were applied. The relationship between a and b is preserved, because these rules force an a always to be followed by a b. Notably, these three simple rewriting rules can be used to generate (or describe) infinitely many "similar" structures. As we show in Chapter 12, this approach also has the advantage of a solid theoretical foundation.

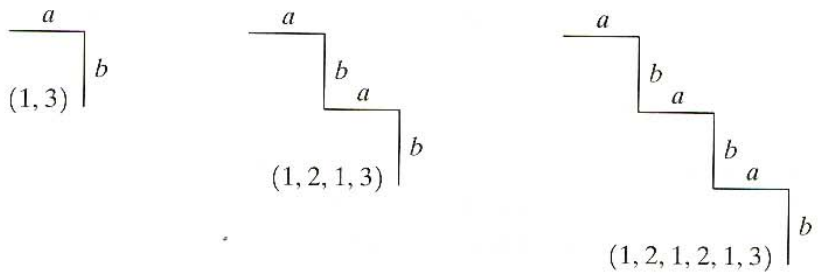


FIGURE 11.31
Sample derivations for the rules $S \rightarrow aA$, $A \rightarrow bS$, and $A \rightarrow b$.

Because strings are 1-D structures, their application to image description requires establishing an appropriate method for reducing 2-D positional relations to 1-D form. Most applications of strings to image description are based on the idea of extracting connected line segments from the objects of interest. One approach is to follow the contour of an object and code the result with segments of specified direction and/or length. Figure 11.32 illustrates this procedure.

Another, somewhat more general, approach is to describe sections of an image (such as small homogeneous regions) by directed line segments, which can be joined in other ways besides head-to-tail connections. Figure 11.33(a) illustrates this approach, and Fig. 11.33(b) shows some typical operations that can be defined on abstracted primitives. Figure 11.33(c) shows a set of specific primitives consisting of line segments defined in four directions, and Fig. 11.33(d) shows a step-by-step generation of a specific shape, where $(\sim d)$ indicates the primitive d with its direction reversed. Note that each composite structure has a single head and a single tail. The result of interest is the last string, which describes the complete structure.

String descriptions are best suited for applications in which connectivity of primitives can be expressed in a head-to-tail or other continuous manner. Sometimes regions that are similar in terms of texture or other descriptor may not be contiguous, and techniques are required for describing such situations. One of the most useful approaches for doing so is to use tree descriptors.

A *tree* T is a finite set of one or more nodes for which

(a) there is a unique node \$ designated the *root*, and
(b) the remaining nodes are partitioned into m disjointed sets $T_1, \ldots, T_m$, each of which in turn is a tree called a *subtree* of T.

The *tree frontier* is the set of nodes at the bottom of the tree (the *leaves*), taken in order from left to right. For example, the tree shown in Fig. 11.34 has root \$ and frontier xy.

Generally, two types of information in a tree are important: (1) information about a node stored as a set of words describing the node, and (2) information relating a node to its neighbors, stored as a set of pointers to those neighbors.

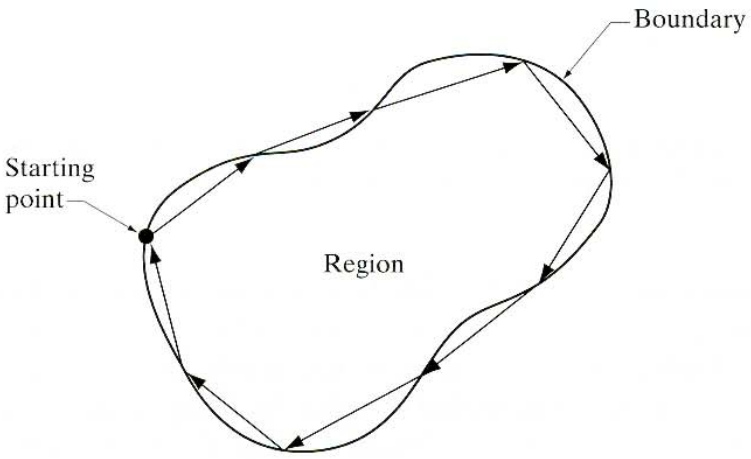
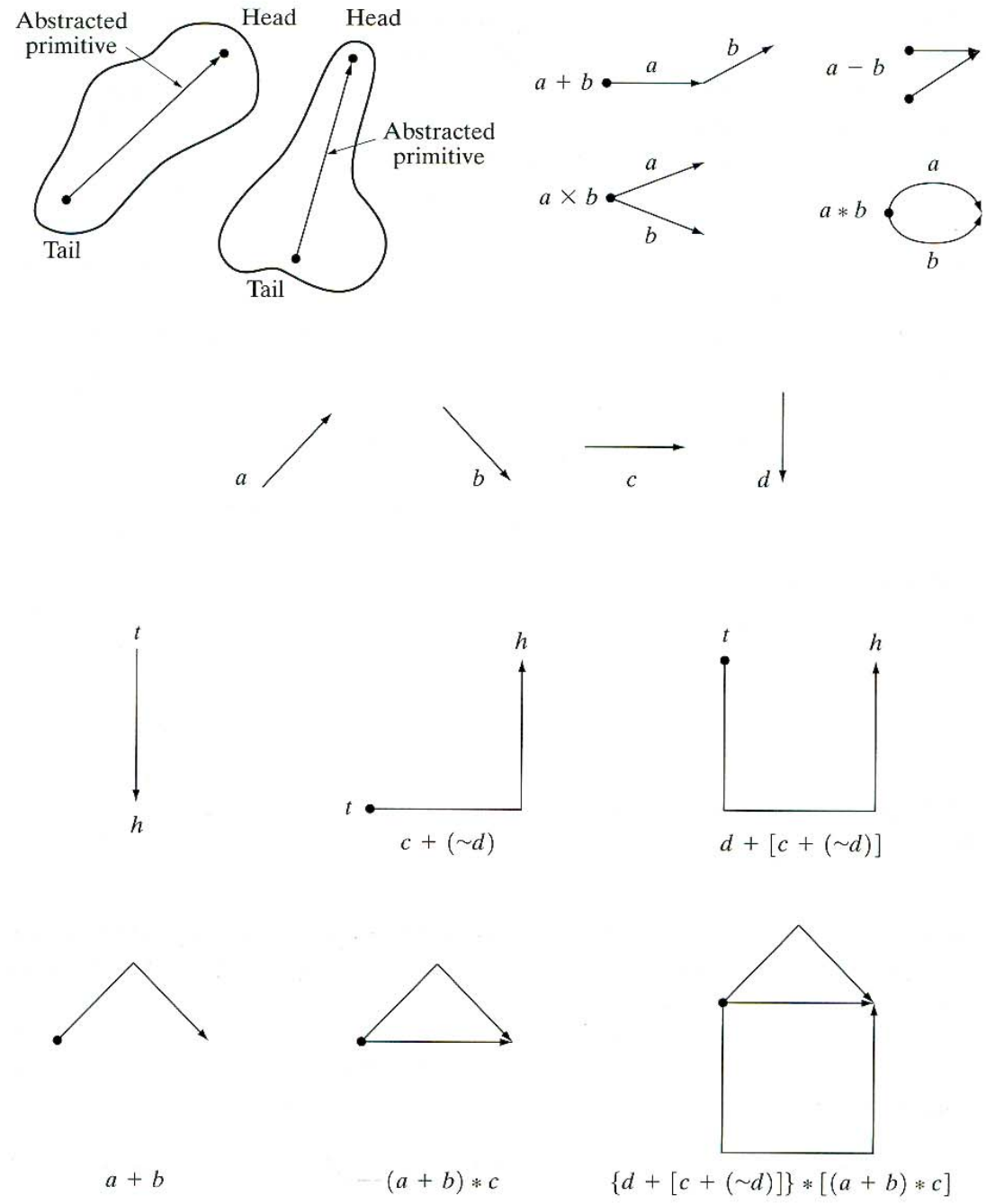


FIGURE 11.32 Coding a region boundary with directed line segments.



a b
c
d

FIGURE 11.33 (a) Abstracted primitives. (b) Operations among primitives. (c) A set of specific primitives. (d) Steps in building a structure.

As used in image description, the first type of information identifies an image substructure (e.g., region or boundary segment), whereas the second type defines the physical relationship of that substructure to other substructures. For example, Fig. 11.35(a) can be represented by a tree by using the relationship "inside of." Thus, if the root of the tree is denoted $, Fig. 11.35(a) shows that the first

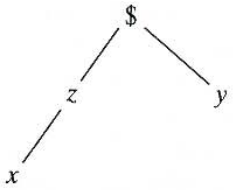
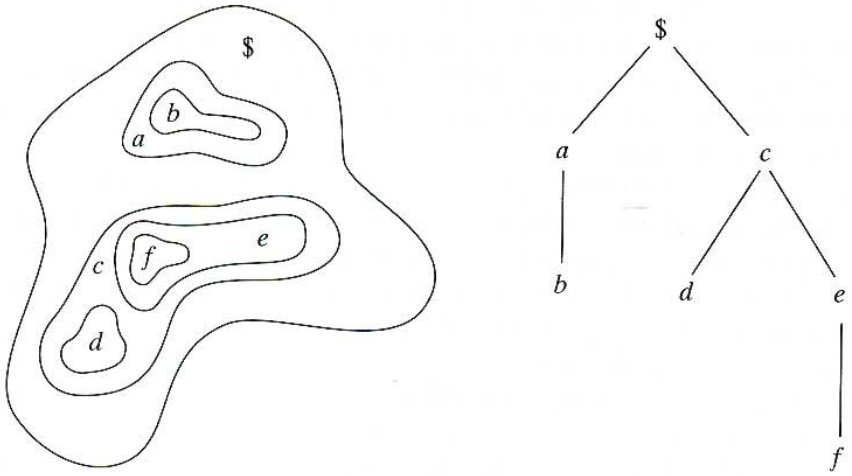


FIGURE 11.34 A simple tree with root $ and frontier xy.



a b

FIGURE 11.35 (a) A simple composite region. (b) Tree representation obtained by using the relationship "inside of."

level of complexity involves a and c inside \$, which produces two branches emanating from the root, as shown in Fig. 11.35(b). The next level involves b inside a, and d and e inside c. Finally, f inside e completes the tree.

Summary

The representation and description of objects or regions that have been segmented out of an image are early steps in the operation of most automated processes involving images. These descriptions, for example, constitute the input to the object recognition methods developed in the following chapter. As indicated by the range of description techniques covered in this chapter, the choice of one method over another is determined by the problem under consideration. The objective is to choose descriptors that "capture" essential differences between objects, or classes of objects, while maintaining as much independence as possible to changes in factors such as location, size, and orientation.

References and Further Reading

The chain-code representation discussed in Section 11.1.1 was first proposed by Freeman [1961, 1974]. For current work using chain codes see Bribiesca [1999], who also has extended chain codes to 3-D (Bribiesca [2000]). For a detailed discussion and algorithm to compute minimum-perimeter polygons (Section 11.1.2) see Sklansky et al. [1972]. Typical

work on polygonal approximations a decade ago is illustrated in the papers by Bengtsoon and Eklundh [1991] and by Sato [1992]. The paper by Zhu and Chirlian [1995] presents an interesting approach to the detection of point inflections along a curve. See also Hu and Yan [1997]. More recent work in this area focuses on invariant polygonal fitting (Voss and Suesse [1997]), on methods for evaluating the performance of polygonal approximation algorithms (Rosin [1997]), on generic implementations (Huang and Sun [1999]), and on computational speed (Davis [1999]).

References for the discussion of signatures (Section 11.1.3) are Ballard and Brown [1982] and Gupta and Srinath [1988]. See Preparata and Shamos [1985] regarding fundamental formulations for finding the convex hull and convex deficiency (Section 11.1.4). See also the paper by Liu-Yu and Antipolis [1993]. Katzir et al. [1994] discuss the detection of partially occluded curves. Zimmer et al. [1997] discuss an improved algorithm for computing the convex hull, and Latecki and Lakämper [1999] discuss a convexity rule for shape decomposition.

The skeletonizing algorithm discussed in Section 11.1.5 is based on Zhang and Suen [1984]. Some useful additional comments on the properties and implementation of this algorithm are included in a paper by Lu and Wang [1986]. A paper by Jang and Chin [1990] provides an interesting tie between the discussion in Section 11.1.5 and the morphological concept of thinning introduced in Section 9.5.5. For thinning approaches in the presence of noise see Shi and Wong [1994] and Chen and Yu [1996]. Shaked and Bruckstein [1998] discuss a pruning algorithm useful for removing spurs from a skeleton. Fast computation of the medial axis transform is discussed by Sahni and Jenq [1992] and by Ferreira and Ubéda [1999]. The survey paper by Loncaric [1998] is of interest regarding many of the approaches discussed in Section 11.1.

Freeman and Shapira [1975] give an algorithm for finding the basic rectangle of a closed, chain-coded curve (Section 11.2.1). The discussion on shape numbers in Section 11.2.2 is based on the work of Bribiesca and Guzman [1980] and Bribiesca [1981]. For additional reading on Fourier descriptors (Section 11.2.3), see the early papers by Zahn and Roskies [1972] and by Persoon and Fu [1977]. See also Aguado et al. [1998] and Sonka et al. [1999]. Reddy and Chatterji [1996] discuss an interesting approach using the FFT to achieve invariance to translation, rotation, and scale change. The material in Section 11.2.4 is based on elementary probability theory (see, for example, Peebles [1993] and Popoulis [1991]).

For additional reading on Section 11.3.2, see Rosenfeld and Kak [1982] and Ballard and Brown [1982]. For an excellent introduction to texture (Section 11.3.3), see Haralick and Shapiro [1992]. For an early survey on texture, see Wechsler [1980]. The papers by Murino et al. [1998] and Garcia [1999], and the discussion by Shapiro and Stockman [2001], are representative of current work in this field.

The moment invariant approach discussed in Section 11.3.4 is from Hu [1962]. Also see Bell [1965]. To get an idea of the range of applications of moment invariants, see Hall [1979] regarding image matching and Cheung and Teoh [1999] regarding the use of moments for describing symmetry. Moment invariants were generalized to n dimensions by Mamistvalov [1998].

Hotelling [1933] was the first to derive and publish the approach that transforms discrete variables into uncorrelated coefficients. He referred to this technique as *the method of principal components*. His paper gives considerable insight into the method and is worth reading. Hotelling's transformation was rediscovered by Kramer and Mathews [1956] and by Huang and Schultheiss [1963]. Principal components are still a basic tool for image description used in numerous applications, as exemplified by Swets and Weng [1996] and by Duda, Heart, and Stork [2001]. References for the material in Section 11.5 are Gonzalez and Thomason [1978] and Fu [1982]. See also Sonka et al. [1999].

Problems

11.1 ★ **(a)** Show that redefining the starting point of a chain code so that the resulting sequence of numbers forms an integer of minimum magnitude makes the code independent of the initial starting point on the boundary.

(b) Find the normalized starting point of the code 11076765543322.

11.2 **(a)** Show that the first difference of a chain code normalizes it to rotation, as explained in Section 11.1.1.

(b) Compute the first difference of the code 01010303033232322121 11.

11.3 ★ **(a)** Show that the rubber-band polygonal approximation approach discussed in Section 11.1.2 yields a polygon with minimum perimeter.

(b) Show that if each cell corresponds to a pixel on the boundary, the maximum possible error in that cell is $\sqrt{2}d$, where d is the minimum possible horizontal or vertical distance between adjacent pixels (i.e., the distance between lines in the sampling grid used to produce the digital image).

11.4 ★ **(a)** Discuss the effect on the resulting polygon if the error threshold is set to zero in the merging method discussed in Section 11.1.2.

(b) What would be the effect on the splitting method?

11.5 ★ **(a)** Plot the signature of a square boundary using the tangent angle method discussed in Section 11.1.3.

(b) Repeat for the slope density function.

Assume that the square is aligned with the x- and y-axes, and let the x-axis be the reference line. Start at the corner closest to the origin.

11.6 Find an expression for the signature of each of the following boundaries, and plot the signatures.

★ **(a)** An equilateral triangle

(b) A rectangle

(c) An ellipse

11.7 Draw the medial axis of

★ **(a)** A circle

★ **(b)** A square

(c) A rectangle

(d) An equilateral triangle

11.8 For each of the figures shown,

★ **(a)** Discuss the action taken at point p by step 1 of the skeletonizing algorithm presented in Section 11.1.5.

(b) Repeat for step 2 of the algorithm. Assume that $p = 1$ in all cases.

1	1	0
1	p	0
1	1	0

0	0	0
1	p	0
0	0	0

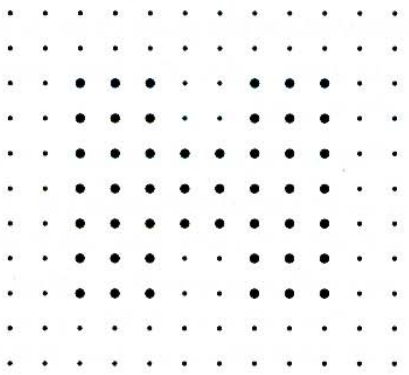
0	1	0
1	p	1
0	1	0

1	1	0
0	p	1
0	0	0

11.9 With reference to the skeletonizing algorithm in Section 11.1.5, what would the figure shown look like after

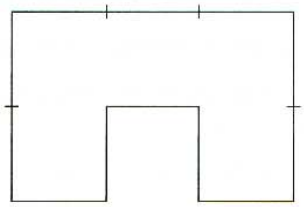
★ **(a)** One pass of step 1 of the algorithm?

(b) One pass of step 2 (on the result of step 1, not the original image)?



11.10 ★ **(a)** What is the order of the shape number for the figure shown?

(b) Obtain the shape number.



11.11 The procedure discussed in Section 11.2.3 for using Fourier descriptors consists of expressing the coordinates of a contour as complex numbers, taking the DFT of these numbers, and keeping only a few components of the DFT as descriptors of the boundary shape. The inverse DFT is then an approximation to the original contour. What class of contour shapes would have a DFT consisting of real numbers and how would the axis system in Fig. 11.13 have to be set up to obtain these real numbers?

★**11.12** Give the smallest number of statistical moment descriptors needed to differentiate between the signatures of the figures shown in Fig. 11.5.

11.13 Give two boundary shapes that have the same mean and third statistical moment descriptors, but different second moments.

★**11.14** Propose a set of descriptors capable of differentiating between the shapes of the characters 0, 1, 8, 9, and X. (*Hint:* Use topological descriptors in conjunction with the convex hull.)

11.15 Consider a checkerboard image composed of alternating black and white squares, each of size $m \times m$. Give a position operator that would yield a diagonal co-occurrence matrix.

11.16 Obtain the gray-level co-occurrence matrix of a 5×5 image composed of a checkerboard of alternating 1's and 0's if

★ **(a)** the position operator P is defined as "one pixel to the right," and

(b) "two pixels to the right."

Assume that the top left pixel has value 0.

11.17 Prove the validity of Eqs. (11.4-7), (11.4-8), and (11.4-9).

★**11.18** It was mentioned in Example 11.10 that a credible job could be done of reconstructing approximations to the six original images by using only the two

principal-component images associated with the largest eigenvalues. What would be the mean square error incurred in doing so? Express your answer as a percentage of the maximum possible error.

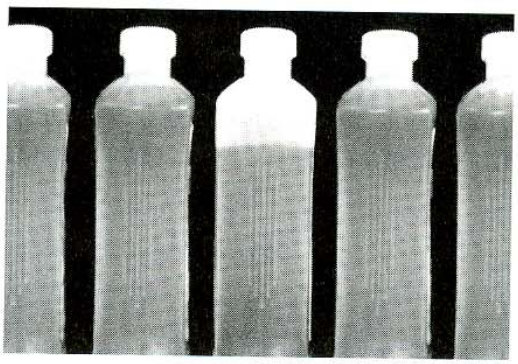
11.19 For a set of images of size 64×64, assume that the covariance matrix given in Eq. (11.4-9) turns out to be the identity matrix. What would be the mean square error between the original images and images reconstructed using Eq. (11.4-11) with only half of the original eigenvectors?

★11.20 Under what conditions would you expect the major axes of a boundary, defined in Section 11.2.1, to be equal to the eigen axes of that boundary?

11.21 Give a spatial relationship and corresponding tree representation for a checkerboard pattern of black and white squares. Assume that the top left element is black and that the root of the tree corresponds to that element. Your tree can have no more than two branches emanating from each node.

★11.22 You are contracted to design an image processing system for detecting imperfections on the inside of certain solid plastic wafers. The wafers are examined using an X-ray imaging system, which yields 8-bit images of 512×512 resolution. In the absence of imperfections, the images appear "bland," having a mean gray level of 100 and variance of 400. The imperfections appear as bloblike regions in which about 70% of the pixels have excursions in intensity of 50 gray levels or less about a mean of 100. A wafer is considered defective if such a region occupies an area exceeding 20×20 pixels in size. Propose a system based on texture analysis.

11.23 A company that bottles a variety of industrial chemicals has heard of your success solving imaging problems and hires you to design an approach for detecting when bottles are not full. The bottles appear as shown in the following figure as they move along a conveyor line past an automatic filling and capping station. A bottle is considered imperfectly filled when the level of the liquid is below the midway point between the bottom of the neck and the shoulder of the bottle. The shoulder is defined as the region of the bottle where the sides and slanted portion of the bottle intersect. The bottles are moving, but the company has an imaging system equipped with a illumination flash front end that effectively stops motion, so you will be given images that look very close to the sample shown here. Based on the material you have learned up to this point, propose a solution for detecting bottles that are not filled properly. State clearly all assumptions that you make and that are likely to impact the solution you propose.



11.24 Having heard about your success with the bottling problem, you are contacted by a fluids company that wishes to automate bubble-counting in certain processes for quality control. The company has solved the imaging problem and can obtain 8-bit images of resolution 700×700 pixels, such as the ones shown. Each

image represents an area of 7 cm^2. The company wishes to do two things with each image: (1) Determine the ratio of the area occupied by bubbles to the total area of the image, and (2) count the number of distinct bubbles. Based on the material you have learned up to this point, propose a solution to this problem. In your solution, make sure to state the physical dimensions of the smallest bubble your solution can detect. State clearly all assumptions that you make and that are likely to impact the solution you propose.