10 Image Segmentation

The whole is equal to the sum of its parts.

Euclid

The whole is greater than the sum of its parts.

Max Wertheimer

Preview

The material in the previous chapter began a transition from image processing methods whose inputs and outputs are images, to methods in which the inputs are images but the outputs are attributes extracted from those images (in the sense defined in Section 1.1). Segmentation is another major step in that direction.

Segmentation subdivides an image into its constituent regions or objects. The level of detail to which the subdivision is carried depends on the problem being solved. That is, segmentation should stop when the objects or regions of interest in an application have been detected. For example, in the automated inspection of electronic assemblies, interest lies in analyzing images of products with the objective of determining the presence or absence of specific anomalies, such as missing components or broken connection paths. There is no point in carrying segmentation past the level of detail required to identify those elements.

Segmentation of nontrivial images is one of the most difficult tasks in image processing. Segmentation accuracy determines the eventual success or failure of computerized analysis procedures. For this reason, considerable care should be taken to improve the probability of accurate segmentation. In some situations, such as in industrial inspection applications, at least some measure of control over the environment typically is possible. The experienced image processing system designer invariably pays considerable attention to such opportunities. In other applications, such as autonomous target acquisition, the system designer has no control over the operating environment, and the usual

approach is to focus on selecting the types of sensors most likely to enhance the objects of interest while diminishing the contribution of irrelevant image detail. A good example is the use of infrared imaging by the military to detect objects with strong heat signatures, such as equipment and troops in motion.

See Sections 6.7 and 10.3.8 for a discussion of segmentation techniques based on more than just gray (intensity) values.

Most of the segmentation algorithms in this chapter are based on one of two basic properties of intensity values: discontinuity and similarity. In the first category, the approach is to partition an image based on abrupt changes in intensity, such as edges. The principal approaches in the second category are based on partitioning an image into regions that are similar according to a set of predefined criteria. Thresholding, region growing, and region splitting and merging are examples of methods in this category. In this chapter, we discuss and illustrate a number of these approaches and show that improvements in segmentation performance can be achieved by combining methods from distinct categories, such as techniques in which edge detection is combined with thresholding. We discuss also image segmentation based on morphology. This approach is particularly attractive because it combines several of the positive attributes of segmentation based on the techniques presented in the first part of the chapter. We conclude the chapter with a brief discussion on the use of motion cues for segmentation.

10.1 Fundamentals

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

(a) $\bigcup_{i=1}^{n} R_i = R$.

See Section 2.5.2 regarding connected sets.

(b) R_i is a connected set, $i = 1, 2, \ldots, n$.

(c) $R_i \cap R_j = \varnothing$ for all i and j, $i \neq j$.

(d) $Q(R_i) = \text{TRUE}$ for $i = 1, 2, \ldots, n$.

(e) $Q(R_i \cup R_j) = \text{FALSE}$ for any adjacent regions R_i and R_j.

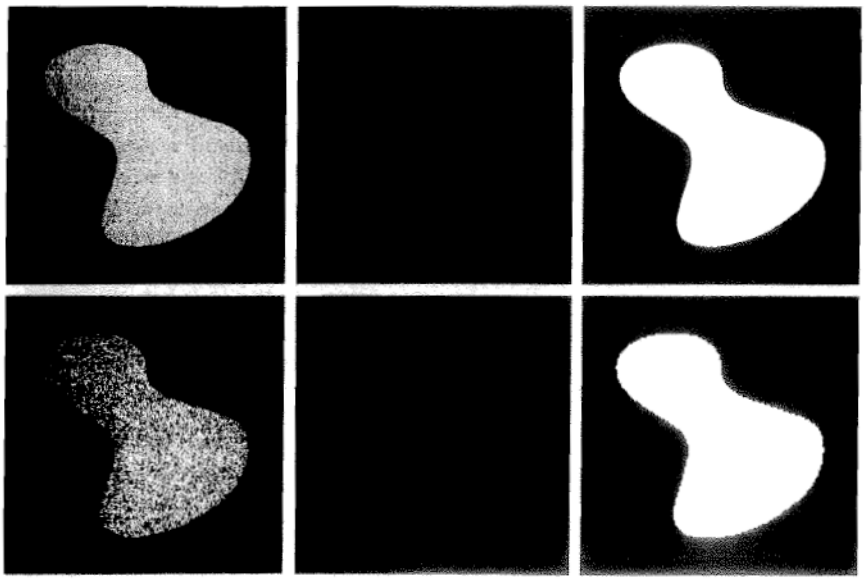
Here, $Q(R_k)$ is a logical predicate defined over the points in set R_k, and $\varnothing$ is the null set. The symbols $\cup$ and $\cap$ represent set union and intersection, respectively, as defined in Section 2.6.4. Two regions R_i and R_j are said to be *adjacent* if their union forms a connected set, as discussed in Section 2.5.2.

Condition (a) indicates that the segmentation must be complete; that is, every pixel must be in a region. Condition (b) requires that points in a region be connected in some predefined sense (e.g., the points must be 4- or 8-connected, as defined in Section 2.5.2). Condition (c) indicates that the regions must be disjoint. Condition (d) deals with the properties that must be satisfied by the pixels in a segmented region—for example, $Q(R_i) = \text{TRUE}$ if all pixels in R_i have the same intensity level. Finally, condition (e) indicates that two adjacent regions R_i and R_j must be different in the sense of predicate Q.[†]

[†]In general, Q can be a compound expression such as, for example, $Q(R_i) = \text{TRUE}$ if the average intensity of the pixels in R_i is less than m_i, AND if the standard deviation of their intensity is greater than σ_i, where m_i and σ_i are specified constants.

Thus, we see that the fundamental problem in segmentation is to partition an image into regions that satisfy the preceding conditions. Segmentation algorithms for monochrome images generally are based on one of two basic categories dealing with properties of intensity values: discontinuity and similarity. In the first category, the assumption is that boundaries of regions are sufficiently different from each other and from the background to allow boundary detection based on local discontinuities in intensity. *Edge-based segmentation* is the principal approach used in this category. *Region-based segmentation* approaches in the second category are based on partitioning an image into regions that are similar according to a set of predefined criteria.

Figure 10.1 illustrates the preceding concepts. Figure 10.1(a) shows an image of a region of constant intensity superimposed on a darker background, also of constant intensity. These two regions comprise the overall image region. Figure 10.1(b) shows the result of computing the boundary of the inner region based on intensity discontinuities. Points on the inside and outside of the boundary are black (zero) because there are no discontinuities in intensity in those regions. To segment the image, we assign one level (say, white) to the pixels on, or interior to, the boundary and another level (say, black) to all points exterior to the boundary. Figure 10.1(c) shows the result of such a procedure. We see that conditions (a) through (c) stated at the beginning of this section are satisfied by



a b c
d e f

FIGURE 10.1 (a) Image containing a region of constant intensity. (b) Image showing the boundary of the inner region, obtained from intensity discontinuities. (c) Result of segmenting the image into two regions. (d) Image containing a textured region. (e) Result of edge computations. Note the large number of small edges that are connected to the original boundary, making it difficult to find a unique boundary using only edge information. (f) Result of segmentation based on region properties.

this result. The predicate of condition (d) is: If a pixel is on, or inside the boundary, label it white; otherwise label it black. We see that this predicate is TRUE for the points labeled black and white in Fig. 10.1(c). Similarly, the two segmented regions (object and background) satisfy condition (e).

The next three images illustrate region-based segmentation. Figure 10.1(d) is similar to Fig. 10.1(a), but the intensities of the inner region form a textured pattern. Figure 10.1(e) shows the result of computing the edges of this image. Clearly, the numerous spurious changes in intensity make it difficult to identify a unique boundary for the original image because many of the nonzero intensity changes are connected to the boundary, so edge-based segmentation is not a suitable approach. We note however, that the outer region is constant, so all we need to solve this simple segmentation problem is a predicate that differentiates between textured and constant regions. The standard deviation of pixel values is a measure that accomplishes this, because it is nonzero in areas of the texture region and zero otherwise. Figure 10.1(f) shows the result of dividing the original image into subregions of size 4×4. Each subregion was then labeled white if the standard deviation of its pixels was positive (i.e., if the predicate was TRUE) and zero otherwise. The result has a "blocky" appearance around the edge of the region because groups of 4×4 squares were labeled with the same intensity. Finally, note that these results also satisfy the five conditions stated at the beginning of this section.

10.2 Point, Line, and Edge Detection

The focus of this section is on segmentation methods that are based on detecting sharp, *local* changes in intensity. The three types of image features in which we are interested are isolated points, lines, and edges. *Edge pixels* are pixels at which the intensity of an image function changes abruptly, and *edges* (or *edge segments*) are sets of connected edge pixels (see Section 2.5.2 regarding connectivity). *Edge detectors* are local image processing methods designed to detect edge pixels. A line may be viewed as an edge segment in which the intensity of the background on either side of the line is either much higher or much lower than the intensity of the line pixels. In fact, as we discuss in the following section and in Section 10.2.4, lines give rise to so-called "roof edges." Similarly, an isolated point may be viewed as a line whose length and width are equal to one pixel.

When we refer to lines, we are referring to thin structures, typically just a few pixels thick. Such lines may correspond, for example, to elements of a digitized architectural drawing or roads in a satellite image.

10.2.1 Background

As we saw in Sections 2.6.3 and 3.5, local averaging smooths an image. Given that averaging is analogous to integration, it should come as no surprise that abrupt, local changes in intensity can be detected using derivatives. For reasons that will become evident shortly, first- and second-order derivatives are particularly well suited for this purpose.

Derivatives of a digital function are defined in terms of differences. There are various ways to approximate these differences but, as explained in Section 3.6.1, we require that any approximation used for a first derivative (1) must be zero in areas of constant intensity; (2) must be nonzero at the onset of an intensity step or ramp; and (3) must be nonzero at points along an intensity

ramp. Similarly, we require that an approximation used for a second derivative (1) must be zero in areas of constant intensity; (2) must be nonzero at the onset *and* end of an intensity step or ramp; and (3) must be zero along intensity ramps. Because we are dealing with digital quantities whose values are finite, the maximum possible intensity change is also finite, and the shortest distance over which a change can occur is between adjacent pixels.

We obtain an approximation to the first-order derivative at point x of a one-dimensional function $f(x)$ by expanding the function $f(x + \Delta x)$ into a Taylor series about x, letting $\Delta x = 1$, and keeping only the linear terms (Problem 10.1). The result is the digital difference

Recall from Section 2.4.2 that increments between image samples are defined as unity for notational clarity, hence the use of $\Delta x = 1$ in the derivation of Eq. (10.2-1).

$$\frac{\partial f}{\partial x} = f'(x) = f(x + 1) - f(x) \tag{10.2-1}$$

We used a partial derivative here for consistency in notation when we consider an image function of two variables, $f(x, y)$, at which time we will be dealing with partial derivatives along the two spatial axes. Clearly, $\partial f/\partial x = df/dx$ when f is a function of only one variable.

We obtain an expression for the second derivative by differentiating Eq. (10.2-1) with respect to x:

$$\begin{aligned}\frac{\partial^2 f}{\partial x^2} = \frac{\partial f'(x)}{\partial x} &= f'(x + 1) - f'(x) \\ &= f(x + 2) - f(x + 1) - f(x + 1) + f(x) \\ &= f(x + 2) - 2f(x + 1) + f(x)\end{aligned}$$

where the second line follows from Eq. (10.2-1). This expansion is about point $x + 1$. Our interest is on the second derivative about point x, so we subtract 1 from the arguments in the preceding expression and obtain the result

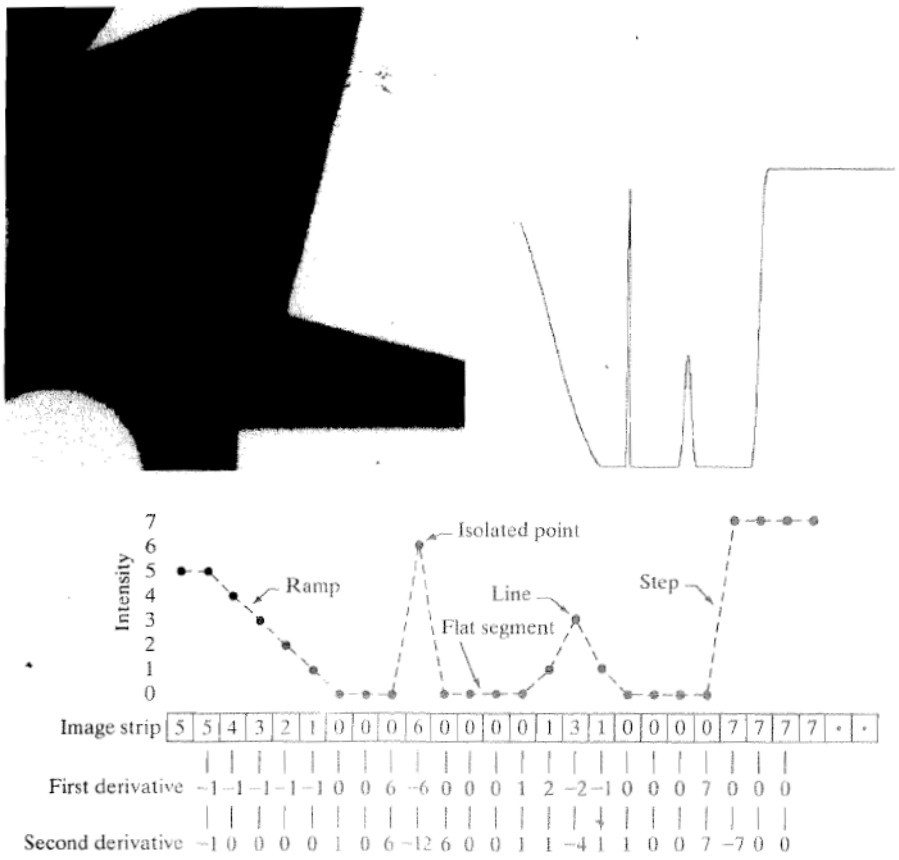
$$\frac{\partial^2 f}{\partial x^2} = f''(x) = f(x + 1) + f(x - 1) - 2f(x) \tag{10.2-2}$$

It easily is verified that Eqs. (10.2-1) and (10.2-2) satisfy the conditions stated at the beginning of this section regarding derivatives of the first and second order. To illustrate this, and also to highlight the fundamental similarities and differences between first- and second-order derivatives in the context of image processing, consider Fig. 10.2.

Figure 10.2(a) shows an image that contains various solid objects, a line, and a single noise point. Figure 10.2(b) shows a horizontal intensity profile (scan line) of the image approximately through its center, including the isolated point. Transitions in intensity between the solid objects and the background along the scan line show two types of edges: *ramp edges* (on the left) and *step edges* (on the right). As we discuss later, intensity transitions involving thin objects such as lines often are referred to as *roof edges*. Figure 10.2(c) shows a simplification of the profile, with just enough points to make it possible for us to analyze numerically how the first- and second-order derivatives behave as they encounter a noise point, a line, and the edges of objects. In this simplified diagram the

transition in the ramp spans four pixels, the noise point is a single pixel, the line is three pixels thick, and the transition of the intensity step takes place between adjacent pixels. The number of intensity levels was limited to eight for simplicity.

Consider the properties of the first and second derivatives as we traverse the profile from left to right. Initially, we note that the first-order derivative is nonzero at the onset and along the entire intensity ramp, while the second-order derivative is nonzero only at the onset and end of the ramp. Because edges of digital images resemble this type of transition, we conclude that first-order derivatives produce "thick" edges and second-order derivatives much finer ones. Next we encounter the isolated noise point. Here, the magnitude of the response at the point is much stronger for the second- than for the first-order derivative. This is not unexpected, because a second-order derivative is much



a b
c

FIGURE 10.2 (a) Image. (b) Horizontal intensity profile through the center of the image, including the isolated noise point. (c) Simplified profile (the points are joined by dashes for clarity). The image strip corresponds to the intensity profile, and the numbers in the boxes are the intensity values of the dots shown in the profile. The derivatives were obtained using Eqs. (10.2-1) and (10.2-2).

more aggressive than a first-order derivative in enhancing sharp changes. Thus, we can expect second-order derivatives to enhance fine detail (including noise) much more than first-order derivatives. The line in this example is rather thin, so it too is fine detail, and we see again that the second derivative has a larger magnitude. Finally, note in both the ramp and step edges that the second derivative has opposite signs (negative to positive or positive to negative) as it transitions into and out of an edge. This "double-edge" effect is an important characteristic that, as we show in Section 10.2.6, can be used to locate edges. The sign of the second derivative is used also to determine whether an edge is a transition from light to dark (negative second derivative) or from dark to light (positive second derivative), where the sign is observed as we move *into* the edge.

In summary, we arrive at the following conclusions: (1) First-order derivatives generally produce thicker edges in an image. (2) Second-order derivatives have a stronger response to fine detail, such as thin lines, isolated points, and noise. (3) Second-order derivatives produce a double-edge response at ramp and step transitions in intensity. (4) The sign of the second derivative can be used to determine whether a transition into an edge is from light to dark or dark to light.

The approach of choice for computing first and second derivatives at every pixel location in an image is to use spatial filters. For the 3×3 filter mask in Fig. 10.3, the procedure is to compute the sum of products of the mask coefficients with the intensity values in the region encompassed by the mask. That is, with reference to Eq. (3.4.3), the *response* of the mask at the center point of the region is

$$\begin{aligned} R &= w_1 z_1 + w_2 z_2 + \cdots + w_9 z_9 \\ &= \sum_{k=1}^{9} w_k z_k \end{aligned} \tag{10.2-3}$$

where z_k is the intensity of the pixel whose spatial location corresponds to the location of the kth coefficient in the mask. The details of implementing this operation over all pixels in an image are discussed in detail in Sections 3.4 and 3.6. In other words, computation of derivatives based on spatial masks is spatial filtering of an image with those masks, as explained in those sections.†

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

FIGURE 10.3
A general 3×3 spatial filter mask.

†As explained in Section 3.4.3, Eq. (10.2-3) is simplified notation either for spatial correlation, given by Eq. (3.4-1), or spatial convolution, given by Eq. (3.4-2). Therefore, when R is evaluated at all locations in an image, the result is an array. All spatial filtering in this chapter is done using correlation. In some instances, we use the term *convolving a mask with an image* as a matter of convention. However, we use this terminology only when the filter masks are symmetric, in which case correlation and convolution yield the same result.

10.2.2 Detection of Isolated Points

Based on the conclusions reached in the preceding section, we know that point detection should be based on the second derivative. From the discussion in Section 3.6.2, this implies using the Laplacian:

$$\nabla^2 f(x, y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{10.2-4}$$

where the partials are obtained using Eq. (10.2-2):

$$\frac{\partial^2 f(x, y)}{\partial x^2} = f(x + 1, y) + f(x - 1, y) - 2f(x, y) \tag{10.2-5}$$

and

$$\frac{\partial^2 f(x, y)}{\partial y^2} = f(x, y + 1) + f(x, y - 1) - 2f(x, y) \tag{10.2-6}$$

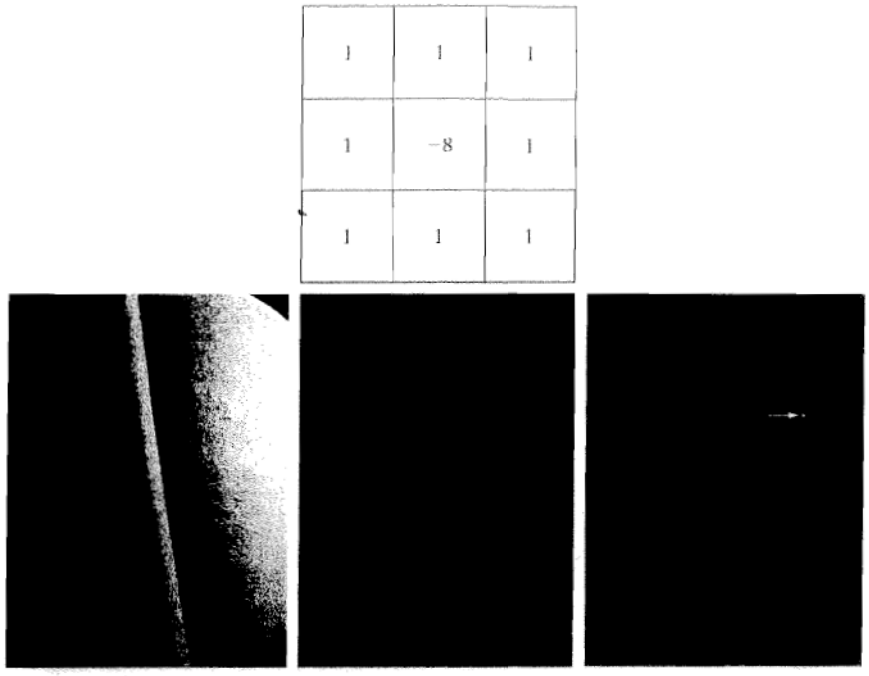
The Laplacian is then

$$\begin{aligned} \nabla^2 f(x, y) = {} & f(x + 1, y) + f(x - 1, y) + f(x, y + 1) \\ & + f(x, y - 1) - 4f(x, y) \end{aligned} \tag{10.2-7}$$

As explained in Section 3.6.2, this expression can be implemented using the mask in Fig. 3.37(a). Also, as explained in that section, we can extend Eq. (10.2-7) to include the diagonal terms, and use the mask in Fig. 3.37(b). Using the Laplacian mask in Fig. 10.4(a), which is the same as the mask in Fig. 3.37(b), we say that a point has been detected at the location (x, y) on which the mask is centered if the absolute value of the response of the mask at that point exceeds a specified threshold. Such points are labeled 1 in the output image and all others are labeled 0, thus producing a binary image. In other words, the output is obtained using the following expression:

$$g(x, y) = \begin{cases} 1 & \text{if } |R(x, y)| \geq T \\ 0 & \text{otherwise} \end{cases} \tag{10.2-8}$$

where g is the output image, T is a nonnegative threshold, and R is given by Eq. (10.2-3). This formulation simply measures the weighted differences between a pixel and its 8-neighbors. Intuitively, the idea is that the intensity of an isolated point will be quite different from its surroundings and thus will be easily detectable by this type of mask. The only differences in intensity that are considered of interest are those large enough (as determined by T) to be considered isolated points. Note that, as usual for a derivative mask, the coefficients sum to zero, indicating that the mask response will be zero in areas of constant intensity.



a
b c d

FIGURE 10.4 (a) Point detection (Laplacian) mask. (b) X-ray image of turbine blade with a porosity. The porosity contains a single black pixel. (c) Result of convolving the mask with the image. (d) Result of using Eq. (10.2-8) showing a single point (the point was enlarged to make it easier to see). (Original image courtesy of X-TEK Systems, Ltd.)

EXAMPLE 10.1: Detection of isolated points in an image.

■ We illustrate segmentation of isolated points in an image with the aid of Fig. 10.4(b), which is an X-ray image of a turbine blade from a jet engine. The blade has a porosity in the upper-right quadrant of the image, and there is a single black pixel embedded within the porosity. Figure 10.4(c) is the result of applying the point detector mask to the X-ray image, and Fig. 10.4(d) shows the result of using Eq. (10.2-8) with T equal to 90% of the highest absolute pixel value of the image in Fig. 10.4(c). The single pixel is clearly visible in this image (the pixel was enlarged manually to enhance its visibility). This type of detection process is rather specialized, because it is based on abrupt intensity changes at single-pixel locations that are surrounded by a homogeneous background in the area of the detector mask. When this condition is not satisfied, other methods discussed in this chapter are more suitable for detecting intensity changes. ■

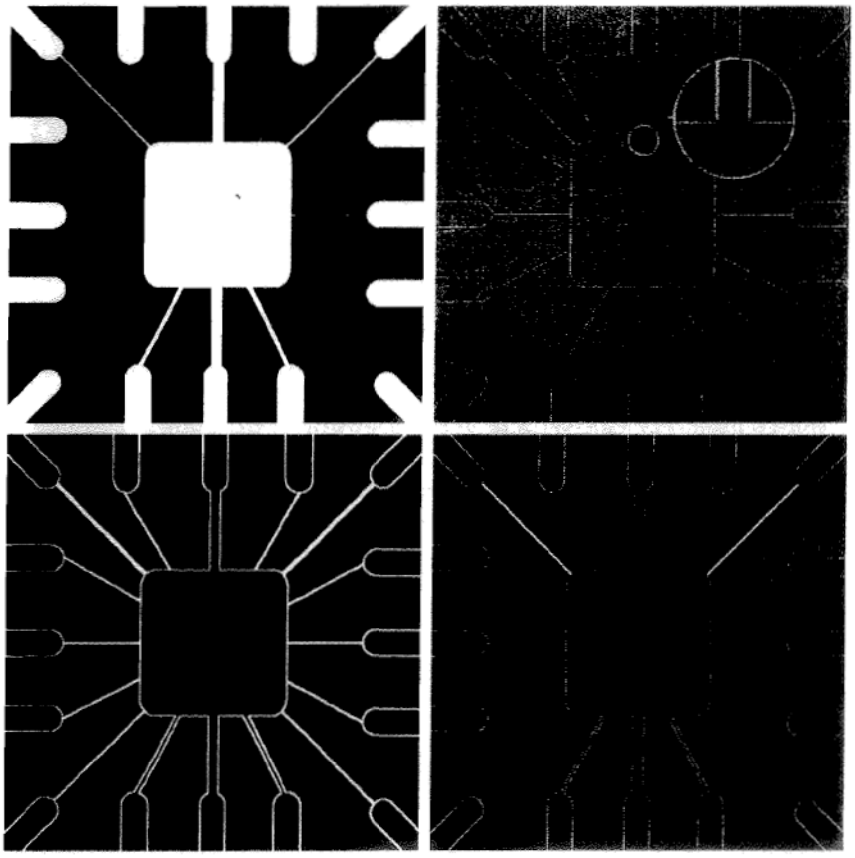
10.2.3 Line Detection

The next level of complexity is line detection. Based on the discussion in Section 10.2.1, we know that for line detection we can expect second derivatives to result in a stronger response and to produce thinner lines than first derivatives. Thus, we can use the Laplacian mask in Fig. 10.4(a) for line detection also, keeping in mind that the double-line effect of the second derivative must be handled properly. The following example illustrates the procedure.

EXAMPLE 10.2: Using the Laplacian for line detection.

Figure 10.5(a) shows a 486 × 486 (binary) portion of a wire-bond mask for an electronic circuit, and Fig. 10.5(b) shows its Laplacian image. Because the Laplacian image contains negative values,† scaling is necessary for display. As the magnified section shows, mid gray represents zero, darker shades of gray represent negative values, and lighter shades are positive. The double-line effect is clearly visible in the magnified region.

At first, it might appear that the negative values can be handled simply by taking the absolute value of the Laplacian image. However, as Fig. 10.5(c) shows, this approach doubles the thickness of the lines. A more suitable approach is to use only the positive values of the Laplacian (in noisy situations we use the values that exceed a positive threshold to eliminate random variations about zero caused by the noise). As the image in Fig. 10.5(d) shows, this approach results in thinner lines, which are considerably more useful. Note in Figs. 10.5(b) through (d) that when the lines are wide with respect to the size of the Laplacian mask, the lines are separated by a zero "valley."



a b
c d

FIGURE 10.5
(a) Original image. (b) Laplacian image; the magnified section shows the positive/negative double-line effect characteristic of the Laplacian. (c) Absolute value of the Laplacian. (d) Positive values of the Laplacian.

†When a mask whose coefficients sum to zero is convolved with an image, the pixels in the resulting image will sum to zero also (Problem 3.16), implying the existence of both positive and negative pixels in the result. Scaling so that all values are nonnegative is required for display purposes.

This is not unexpected. For example, when the 3×3 filter is centered on a line of constant intensity 5 pixels wide, the response will be zero, thus producing the effect just mentioned. When we talk about line detection, the assumption is that lines are thin with respect to the size of the detector. Lines that do not satisfy this assumption are best treated as regions and handled by the edge detection methods discussed later in this section. ■

The Laplacian detector in Fig. 10.4(a) is isotropic, so its response is independent of direction (with respect to the four directions of the 3×3 Laplacian mask: vertical, horizontal, and two diagonals). Often, interest lies in detecting lines in *specified* directions. Consider the masks in Fig. 10.6. Suppose that an image with a constant background and containing various lines (oriented at 0°, ±45°, and 90°) is filtered with the first mask. The maximum responses would occur at image locations in which a horizontal line passed through the middle row of the mask. This is easily verified by sketching a simple array of 1s with a line of a different intensity (say, 5s) running horizontally through the array. A similar experiment would reveal that the second mask in Fig. 10.6 responds best to lines oriented at +45°; the third mask to vertical lines; and the fourth mask to lines in the −45° direction. The preferred direction of each mask is weighted with a larger coefficient (i.e., 2) than other possible directions. The coefficients in each mask sum to zero, indicating a zero response in areas of constant intensity.

Let R_1, R_2, R_3, and R_4 denote the responses of the masks in Fig. 10.6, from left to right, where the Rs are given by Eq. (10.2-3). Suppose that an image is filtered (individually) with the four masks. If, at a given point in the image, $|R_k| > |R_j|$, for all $j \neq k$, that point is said to be more likely associated with a line in the direction of mask k. For example, if at a point in the image, $|R_1| > |R_j|$ for $j = 2, 3, 4$, that particular point is said to be more likely associated with a horizontal line. Alternatively, we may be interested in detecting lines in a specified direction. In this case, we would use the mask associated with that direction and threshold its output, as in Eq. (10.2-8). In other words, if we are interested in detecting all the lines in an image in the direction defined by a given mask, we simply run the mask through the image and threshold the absolute value of the result. The points that are left are the strongest responses which, for lines 1 pixel thick, correspond closest to the direction defined by the mask. The following example illustrates this procedure.

Recall from Section 2.4.2 that the image axis convention has the origin at the top left, with the positive x-axis pointing down and the positive y-axis extending to the right. The angles of the lines discussed in this section are measured with respect to the positive x-axis. For example, a vertical line has an angle of 0°, and a +45° line extends downward and to the right.

Do not confuse our use of R to designate mask response with the same symbol to denote regions in Section 10.1.

−1	−1	−1	2	−1	−1	−1	2	−1	−1	−1	2
2	2	2	−1	2	−1	−1	2	−1	−1	2	−1
−1	−1	−1	−1	−1	2	−1	2	−1	2	−1	−1
Horizontal			+45°			Vertical			−45°		

FIGURE 10.6 Line detection masks. Angles are with respect to the axis system in Fig. 2.18(b).

EXAMPLE 10.3: Detection of lines in specified directions.

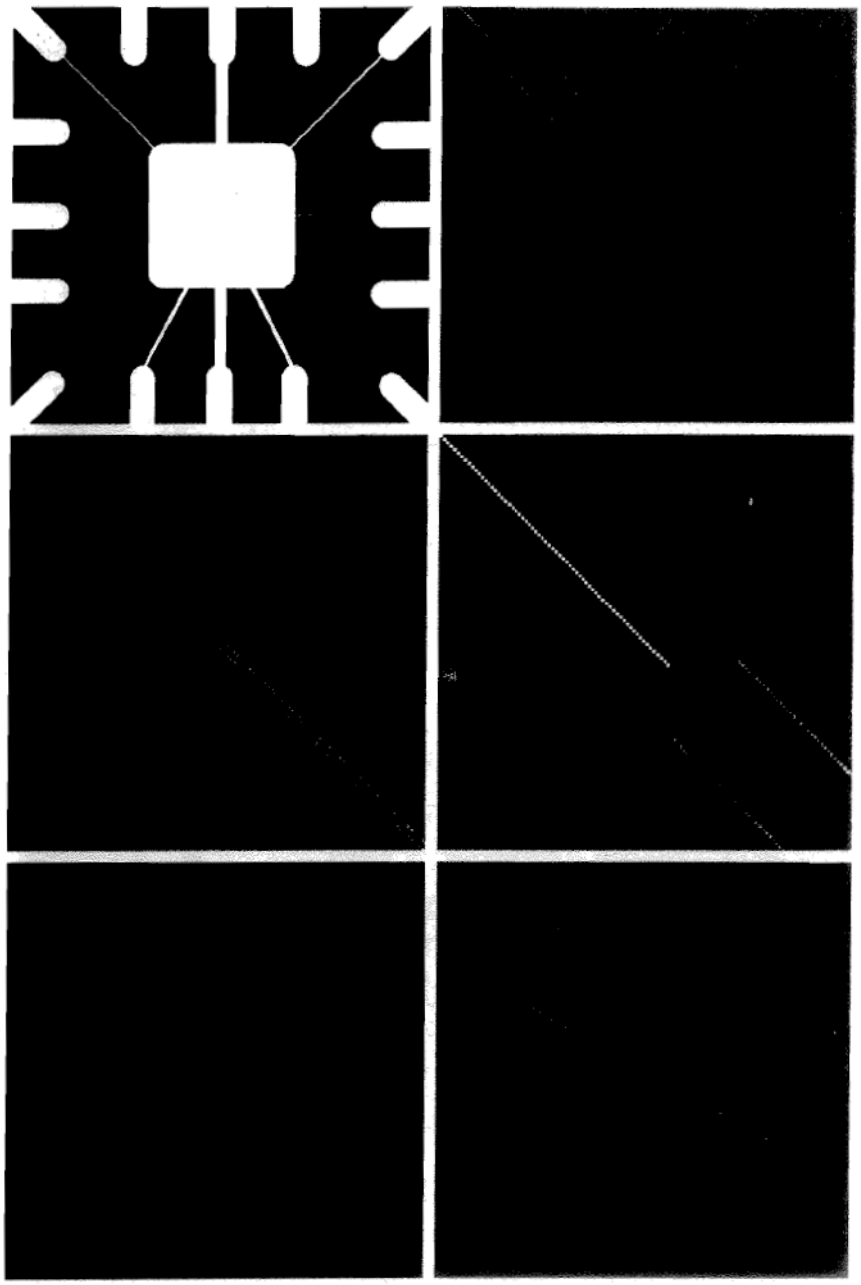
■ Figure 10.7(a) shows the image used in the previous example. Suppose that we are interested in finding all the lines that are 1 pixel thick and oriented at +45°. For this purpose, we use the second mask in Fig. 10.6. Figure 10.7(b) is the result of filtering the image with that mask. As before, the shades darker than the gray background in Fig. 10.7(b) correspond to negative values. There are two principal segments in the image oriented in the +45° direction, one at the top left and one at the bottom right. Figures 10.7(c) and (d) show zoomed sections of Fig. 10.7(b) corresponding to these two areas. Note how much brighter the straight line segment in Fig. 10.7(d) is than the segment in Fig. 10.7(c). The reason is that the line segment in the bottom right of Fig. 10.7(a) is 1 pixel thick, while the one at the top left is not. The mask is "tuned" to detect 1-pixel-thick lines in the +45° direction, so we expect its response to be stronger when such lines are detected. Figure 10.7(e) shows the positive values of Fig. 10.7(b). Because we are interested in the strongest response, we let T equal the maximum value in Fig. 10.7(e). Figure 10.7(f) shows in white the points whose values satisfied the condition $g \geq T$, where g is the image in Fig. 10.7(e). The isolated points in the figure are points that also had similarly strong responses to the mask. In the original image, these points and their immediate neighbors are oriented in such a way that the mask produced a maximum response at those locations. These isolated points can be detected using the mask in Fig. 10.4(a) and then deleted, or they can be deleted using morphological operators, as discussed in the last chapter. ■

10.2.4 Edge Models

Edge detection is the approach used most frequently for segmenting images based on abrupt (local) changes in intensity. We begin by introducing several ways to model edges and then discuss a number of approaches for edge detection.

Edge models are classified according to their intensity profiles. A *step edge* involves a transition between two intensity levels occurring ideally over the distance of 1 pixel. Figure 10.8(a) shows a section of a vertical step edge and a horizontal intensity profile through the edge. Step edges occur, for example, in images generated by a computer for use in areas such as solid modeling and animation. These clean, *ideal* edges can occur over the distance of 1 pixel, provided that no additional processing (such as smoothing) is used to make them look "real." Digital step edges are used frequently as edge models in algorithm development. For example, the Canny edge detection algorithm discussed in Section 10.2.6 was derived using a step-edge model.

In practice, digital images have edges that are blurred and noisy, with the degree of blurring determined principally by limitations in the focusing mechanism (e.g., lenses in the case of optical images), and the noise level determined principally by the electronic components of the imaging system. In such situations, edges are more closely modeled as having an intensity *ramp* profile, such as the edge in Fig. 10.8(b). The slope of the ramp is inversely proportional to the degree of blurring in the edge. In this model, we no longer have a thin (1 pixel thick) path. Instead, an edge point now is any point contained in the ramp, and an edge segment would then be a set of such points that are connected.



a b
c d
e f

FIGURE 10.7
(a) Image of a wire-bond template. (b) Result of processing with the +45° line detector mask in Fig. 10.6. (c) Zoomed view of the top left region of (b). (d) Zoomed view of the bottom right region of (b). (e) The image in (b) with all negative values set to zero. (f) All points (in white) whose values satisfied the condition $g \geq T$, where g is the image in (e). (The points in (f) were enlarged to make them easier to see.)

A third model of an edge is the so-called *roof edge*, having the characteristics illustrated in Fig. 10.8(c). Roof edges are models of lines through a region, with the base (width) of a roof edge being determined by the thickness and sharpness of the line. In the limit, when its base is 1 pixel wide, a roof edge is

a b c

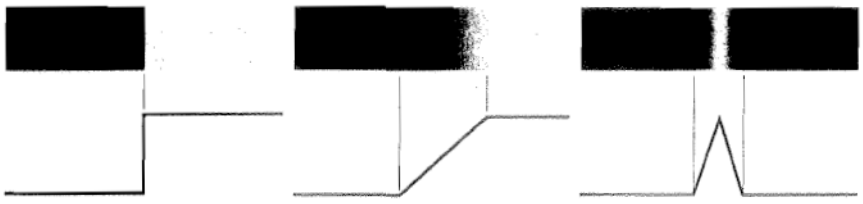


FIGURE 10.8
From left to right, models (ideal representations) of a step, a ramp, and a roof edge, and their corresponding intensity profiles.

really nothing more than a 1-pixel-thick line running through a region in an image. Roof edges arise, for example, in range imaging, when thin objects (such as pipes) are closer to the sensor than their equidistant background (such as walls). The pipes appear brighter and thus create an image similar to the model in Fig. 10.8(c). As mentioned earlier, other areas in which roof edges appear routinely are in the digitization of line drawings and also in satellite images, where thin features, such as roads, can be modeled by this type of edge.

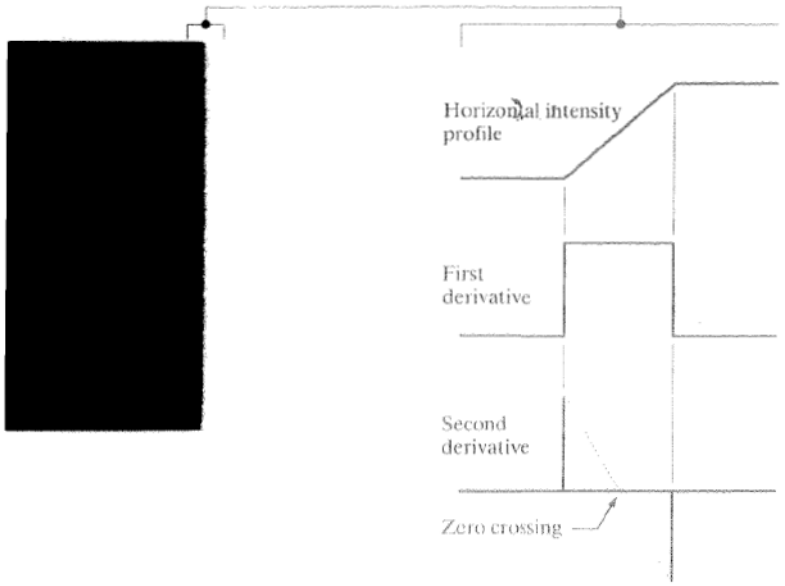
It is not unusual to find images that contain all three types of edges. Although blurring and noise result in deviations from the ideal shapes, edges in images that are reasonably sharp and have a moderate amount of noise do *resemble* the characteristics of the edge models in Fig. 10.8, as the profiles in Fig. 10.9 illustrate.† What the models in Fig. 10.8 allow us to do is write mathematical expressions for edges in the development of image processing algorithms. The performance of these algorithms will depend on the differences between actual edges and the models used in developing the algorithms.

FIGURE 10.9 A 1508 × 1970 image showing (zoomed) actual ramp (bottom, left), step (top, right), and roof edge profiles. The profiles are from dark to light, in the areas indicated by the short line segments shown in the small circles. The ramp and "step" profiles span 9 pixels and 2 pixels, respectively. The base of the roof edge is 3 pixels. (Original image courtesy of Dr. David R. Pickens, Vanderbilt University.)

†Ramp edges with a sharp slope of a few pixels often are treated as step edges in order to differentiate them from ramps in the same image whose slopes are more gradual.

Figure 10.10(a) shows the image from which the segment in Fig. 10.8(b) was extracted. Figure 10.10(b) shows a horizontal intensity profile. This figure shows also the first and second derivatives of the intensity profile. As in the discussion in Section 10.2.1, moving from left to right along the intensity profile, we note that the first derivative is positive at the onset of the ramp and at points on the ramp, and it is zero in areas of constant intensity. The second derivative is positive at the beginning of the ramp, negative at the end of the ramp, zero at points on the ramp, and zero at points of constant intensity. The signs of the derivatives just discussed would be reversed for an edge that transitions from light to dark. The intersection between the zero intensity axis and a line extending between the extrema of the second derivative marks a point called the *zero crossing* of the second derivative.

We conclude from these observations that the *magnitude* of the first derivative can be used to detect the presence of an edge at a point in an image. Similarly, the *sign* of the second derivative can be used to determine whether an edge pixel lies on the dark or light side of an edge. We note two additional properties of the second derivative around an edge: (1) it produces two values for every edge in an image (an undesirable feature); and (2) its zero crossings can be used for locating the centers of thick edges, as we show later in this section. Some edge models make use of a smooth transition into and out of the ramp (Problem 10.7). However, the conclusions reached using those models are the same as with an ideal ramp, and working with the latter simplifies theoretical formulations. Finally, although attention thus far has been limited to a 1-D horizontal profile, a similar argument applies to an edge of any orientation in an image. We simply define a profile perpendicular to the edge direction at any desired point and interpret the results in the same manner as for the vertical edge just discussed.



a b

FIGURE 10.10
(a) Two regions of constant intensity separated by an ideal vertical ramp edge. (b) Detail near the edge, showing a horizontal intensity profile, together with its first and second derivatives.

EXAMPLE 10.4: Behavior of the first and second derivatives of a noisy edge.

■ The edges in Fig. 10.8 are free of noise. The image segments in the first column in Fig. 10.11 show close-ups of four ramp edges that transition from a black region on the left to a white region on the right (keep in mind that the entire transition from black to white is a single edge). The image segment at the top left is free of noise. The other three images in the first column are corrupted by additive Gaussian noise with zero mean and standard deviation of 0.1, 1.0, and 10.0 intensity levels, respectively. The graph below each image is a horizontal intensity profile passing through the center of the image. All images have 8 bits of intensity resolution, with 0 and 255 representing black and white, respectively.

Computation of the derivatives for the entire image segment is discussed in the following section. For now, our interest lies on analyzing just the intensity profiles.

Consider the image at the top of the center column. As discussed in connection with Fig. 10.10(b), the derivative of the scan line on the left is zero in the constant areas. These are the two black bands shown in the derivative image. The derivatives at points on the ramp are constant and equal to the slope of the ramp. These constant values in the derivative image are shown in gray. As we move down the center column, the derivatives become increasingly different from the noiseless case. In fact, it would be difficult to associate the last profile in the center column with the first derivative of a ramp edge. What makes these results interesting is that the noise is almost invisible in the images on the left column. These examples are good illustrations of the sensitivity of derivatives to noise.

As expected, the second derivative is even more sensitive to noise. The second derivative of the noiseless image is shown at the top of the right column. The thin white and black vertical lines are the positive and negative components of the second derivative, as explained in Fig. 10.10. The gray in these images represents zero (as discussed earlier, scaling causes zero to show as gray). The only noisy second derivative image that barely resembles the noiseless case is the one corresponding to noise with a standard deviation of 0.1. The remaining second-derivative images and profiles clearly illustrate that it would be difficult indeed to detect their positive and negative components, which are the truly useful features of the second derivative in terms of edge detection.

The fact that such little visual noise can have such a significant impact on the two key derivatives used for detecting edges is an important issue to keep in mind. In particular, image smoothing should be a serious consideration prior to the use of derivatives in applications where noise with levels similar to those we have just discussed is likely to be present. ■

We conclude this section by noting that there are three fundamental steps performed in edge detection:

1. *Image smoothing for noise reduction.* The need for this step is amply illustrated by the results in the second and third columns of Fig. 10.11.
2. *Detection of edge points.* As mentioned earlier, this is a local operation that extracts from an image all points that are potential candidates to become edge points.
3. *Edge localization.* The objective of this step is to select from the candidate edge points only the points that are true members of the set of points comprising an edge.

The remainder of this section deals with techniques for achieving these objectives.

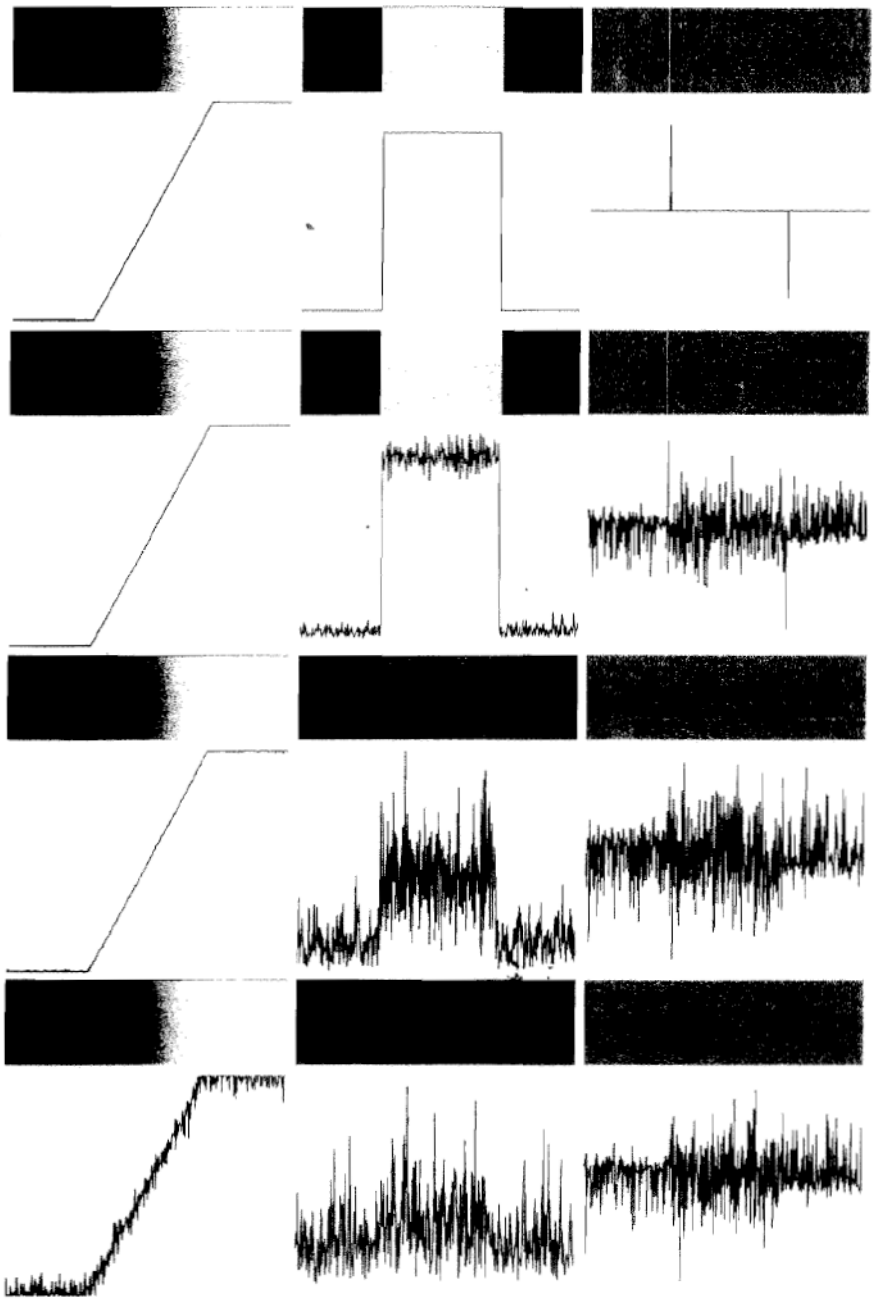


FIGURE 10.11 First column: Images and intensity profiles of a ramp edge corrupted by random Gaussian noise of zero mean and standard deviations of 0.0, 0.1, 1.0, and 10.0 intensity levels, respectively. Second column: First-derivative images and intensity profiles. Third column: Second-derivative images and intensity profiles.

10.2.5 Basic Edge Detection

As illustrated in the previous section, detecting changes in intensity for the purpose of finding edges can be accomplished using first- or second-order derivatives. We discuss first-order derivatives in this section and work with second-order derivatives in Section 10.2.6.

The image gradient and its properties

The tool of choice for finding edge strength *and* direction at location (x, y) of an image, f, is the gradient, denoted by ∇f, and defined as the *vector*

For convenience, we repeat here some equations from Section 3.6.4.

$$\nabla f \equiv \text{grad}(f) \equiv \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f}{\partial x} \\ \dfrac{\partial f}{\partial y} \end{bmatrix} \tag{10.2-9}$$

This vector has the important geometrical property that it points in the direction of the greatest rate of change of f at location (x, y).

The *magnitude* (*length*) of vector ∇f, denoted as $M(x, y)$, where

$$M(x, y) = \text{mag}(\nabla f) = \sqrt{g_x^2 + g_y^2} \tag{10.2-10}$$

is the *value* of the rate of change in the direction of the gradient vector. Note that g_x, g_y, and $M(x, y)$ are images of the same size as the original, created when x and y are allowed to vary over all pixel locations in f. It is common practice to refer to the latter image as the *gradient image*, or simply as the *gradient* when the meaning is clear. The summation, square, and square root operations are *array operations*, as defined in Section 2.6.1.

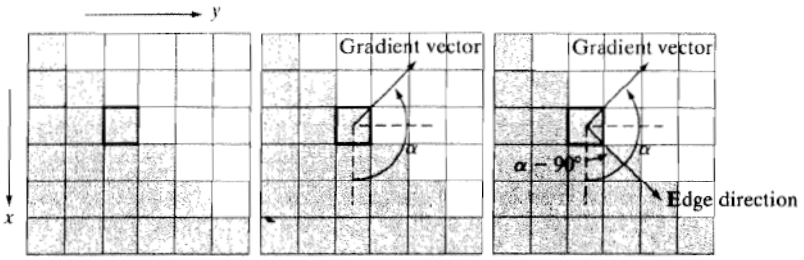
The *direction* of the gradient vector is given by the angle

$$\alpha(x, y) = \tan^{-1}\left[\frac{g_y}{g_x}\right] \tag{10.2-11}$$

measured with respect to the x-axis. As in the case of the gradient image, $\alpha(x, y)$ also is an image of the same size as the original created by the array division of image g_y by image g_x. The direction of an edge at an arbitrary point (x, y) is *orthogonal* to the direction, $\alpha(x, y)$, of the gradient vector at the point.

EXAMPLE 10.5: Properties of the gradient.

■ Figure 10.12(a) shows a zoomed section of an image containing a straight edge segment. Each square shown corresponds to a pixel, and we are interested in obtaining the strength and direction of the edge at the point highlighted with a box. The pixels in gray have value 0 and the pixels in white have value 1. We show following this example that an approach for computing the derivatives in the x- and y-directions using a 3×3 neighborhood centered about a point consists simply of subtracting the pixels in the top row of the neighborhood from the pixels in the bottom row to obtain the partial derivative in the x-direction. Similarly, we subtract the pixels in the left column from the pixels in the right column to obtain the partial derivative in the y-direction. It then



a b c

FIGURE 10.12 Using the gradient to determine edge strength and direction at a point. Note that the edge is perpendicular to the direction of the gradient vector at the point where the gradient is computed. Each square in the figure represents one pixel.

follows, using these differences as our estimates of the partials, that $\partial f/\partial x = -2$ and $\partial f/\partial y = 2$ at the point in question. Then,

Recall from Section 2.4.2 that the origin of the image coordinate system is at the top left, with the positive x- and y-axes extending down and to the right, respectively.

$$\nabla f = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f}{\partial x} \\ \dfrac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} -2 \\ 2 \end{bmatrix}$$

from which we obtain $M(x, y) = 2\sqrt{2}$ at that point. Similarly, the direction of the gradient vector at the same point follows from Eq. (10.2-11): $\alpha(x, y) = \tan^{-1}(g_y/g_x) = -45°$, which is the same as 135° measured in the positive direction with respect to the x-axis. Figure 10.12(b) shows the gradient vector and its direction angle.

Figure 10.12(c) illustrates the important fact mentioned earlier that the edge at a point is orthogonal to the gradient vector at that point. So the direction angle of the edge in this example is $\alpha - 90° = 45°$. All edge points in Fig. 10.12(a) have the same gradient, so the entire edge segment is in the same direction. The gradient vector sometimes is called the *edge normal*. When the vector is normalized to unit length by dividing it by its magnitude [Eq. (10.2-10)], the resulting vector is commonly referred to as the *edge unit normal*. ■

Gradient operators

Obtaining the gradient of an image requires computing the partial derivatives $\partial f/\partial x$ and $\partial f/\partial y$ at every pixel location in the image. We are dealing with digital quantities, so a digital approximation of the partial derivatives over a neighborhood about a point is required. From Section 10.2.1 we know that

$$g_x = \frac{\partial f(x, y)}{\partial x} = f(x + 1, y) - f(x, y) \qquad (10.2\text{-}12)$$

and

$$g_y = \frac{\partial f(x, y)}{\partial y} = f(x, y + 1) - f(x, y) \qquad (10.2\text{-}13)$$

a b

FIGURE 10.13 One-dimensional masks used to implement Eqs. (10.2-12) and (10.2-13).

These two equations can be implemented for all pertinent values of x and y by filtering $f(x, y)$ with the 1-D masks in Fig. 10.13.

When diagonal edge direction is of interest, we need a 2-D mask. The *Roberts cross-gradient operators* (Roberts [1965]) are one of the earliest attempts to use 2-D masks with a diagonal preference. Consider the 3 × 3 region in Fig. 10.14(a). The Roberts operators are based on implementing the diagonal differences

In the remainder of this section we assume implicitly that f is a function of two variables, and omit the variables to simplify the notation.

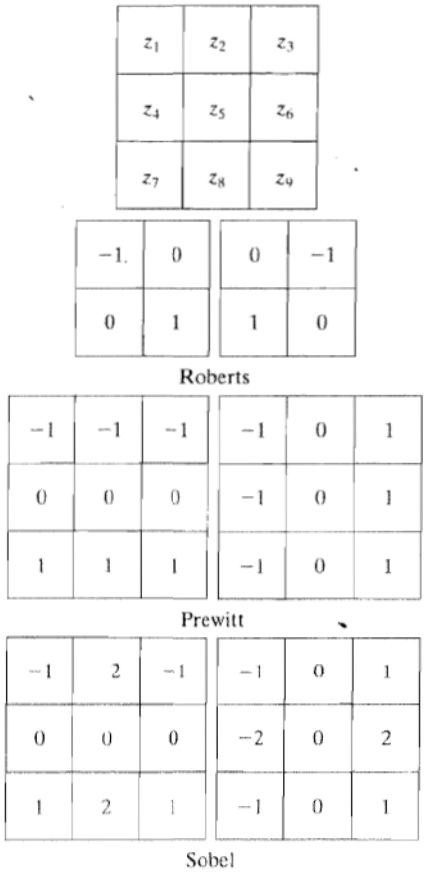
$$g_x = \frac{\partial f}{\partial x} = (z_9 - z_5) \tag{10.2-14}$$

and

$$g_y = \frac{\partial f}{\partial y} = (z_8 - z_6) \tag{10.2-15}$$

a
b c
d e
f g

FIGURE 10.14 A 3 × 3 region of an image (the z's are intensity values) and various masks used to compute the gradient at the point labeled z_5.



Filter masks used to compute the derivatives needed for the gradient are often called *gradient operators, difference operators, edge operators*, or *edge detectors*.

These derivatives can be implemented by filtering an image with the masks in Figs. 10.14(b) and (c).

Masks of size 2×2 are simple conceptually, but they are not as useful for computing edge direction as masks that are symmetric about the center point, the smallest of which are of size 3×3. These masks take into account the nature of the data on opposite sides of the center point and thus carry more information regarding the direction of an edge. The simplest digital approximations to the partial derivatives using masks of size 3×3 are given by

$$g_x = \frac{\partial f}{\partial x} = (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3) \tag{10.2-16}$$

Although these equations encompass a larger neighborhood, we are still dealing with *differences* between intensity values, so the conclusions from earlier discussions regarding first-order derivatives still apply.

and

$$g_y = \frac{\partial f}{\partial y} = (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7) \tag{10.2-17}$$

In these formulations, the difference between the third and first *rows* of the 3×3 region approximates the derivative in the x-direction, and the difference between the third and first columns approximate the derivate in the y-direction. Intuitively, we would expect these approximations to be more accurate than the approximations obtained using the Roberts operators. Equations (10.2-16) and (10.2-17) can be implemented over an entire image by filtering f with the two masks in Figs. 10.14(d) and (e). These masks are called the *Prewitt operators* (Prewitt [1970]).

A slight variation of the preceding two equations uses a weight of 2 in the center coefficient:

$$g_x = \frac{\partial f}{\partial x} = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3) \tag{10.2-18}$$

and

$$g_y = \frac{\partial f}{\partial y} = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7) \tag{10.2-19}$$

It can be shown (Problem 10.10) that using a 2 in the center location provides image smoothing. Figures 10.14(f) and (g) show the masks used to implement Eqs. (10.2-18) and (10.2-19). These masks are called the *Sobel operators* (Sobel [1970]).

The Prewitt masks are simpler to implement than the Sobel masks, but, the slight computational difference between them typically is not an issue. The fact that the Sobel masks have better noise-suppression (smoothing) characteristics makes them preferable because, as mentioned in the previous section, noise suppression is an important issue when dealing with derivatives. Note that the coefficients of all the masks in Fig. 10.14 sum to zero, thus giving a response of zero in areas of constant intensity, as expected of a derivative operator.

a b
c d

FIGURE 10.15
Prewitt and Sobel masks for detecting diagonal edges.

0	1	1	−1	−1	0
−1	0	1	−1	0	1
−1	−1	0	0	1	1

Prewitt

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

Sobel

The masks just discussed are used to obtain the gradient components g_x and g_y at every pixel location in an image. These two partial derivatives are then used to estimate edge strength and direction. Computing the magnitude of the gradient requires that g_x and g_y be combined in the manner shown in Eq. (10.2-10). However, this implementation is not always desirable because of the computational burden required by squares and square roots. An approach used frequently is to approximate the magnitude of the gradient by absolute values:

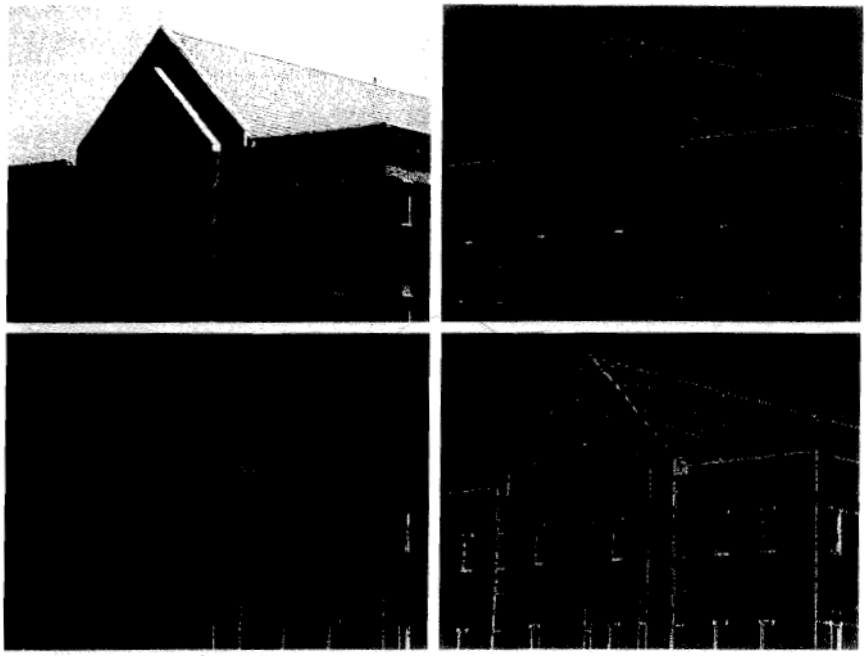
$$M(x, y) \approx |g_x| + |g_y| \tag{10.2-20}$$

This equation is more attractive computationally, and it still preserves relative changes in intensity levels. The price paid for this advantage is that the resulting filters will not be isotropic (invariant to rotation) in general. However, this is not an issue when masks such as the Prewitt and Sobel masks are used to compute g_x and g_y, because these masks give isotropic results only for vertical and horizontal edges. Results would be isotropic only for edges in those two directions, regardless of which of the two equations is used. In addition, Eqs. (10.2-10) and (10.2-20) give identical results for vertical and horizontal edges when the Sobel or Prewitt masks are used (Problem 10.8).

It is possible to modify the 3×3 masks in Fig. 10.14 so that they have their strongest responses along the diagonal directions. Figure 10.15 shows the two additional Prewitt and Sobel masks needed for detecting edges in the diagonal directions.

EXAMPLE 10.6: Illustration of the 2-D gradient magnitude and angle.

Figure 10.16 illustrates the absolute value response of the two components of the gradient, $|g_x|$ and $|g_y|$, as well as the gradient image formed from the sum of these two components. The directionality of the horizontal and vertical components of the gradient is evident in Figs. 10.16(b) and (c). Note, for example, how strong the roof tile, horizontal brick joints, and horizontal segments of the windows are in Fig. 10.16(b) compared to other edges. By contrast.



a b
c d

FIGURE 10.16
(a) Original image of size 834 × 1114 pixels, with intensity values scaled to the range [0, 1]. (b) $|g_x|$, the component of the gradient in the x-direction, obtained using the Sobel mask in Fig. 10.14(f) to filter the image. (c) $|g_y|$, obtained using the mask in Fig. 10.14(g). (d) The gradient image, $|g_x| + |g_y|$.

Fig. 10.16(c) favors features such as the vertical components of the façade and windows. It is common terminology to use the term *edge map* when referring to an image whose principal features are edges, such as gradient magnitude images. The intensities of the image in Fig. 10.16(a) were scaled to the range [0, 1]. We use values in this range to simplify parameter selection in the various methods for edge detection discussed in this section.

Figure 10.17 shows the gradient angle image computed using Eq. (10.2-11). In general, angle images are not as useful as gradient magnitude images for edge detection, but they do complement the information extracted from an image using the magnitude of the gradient. For instance, the constant intensity areas in Fig. 10.16(a), such as the front edge of the sloping roof and top horizontal bands of the front wall, are constant in Fig. 10.17, indicating that the gradient vector direction at all the pixel locations in those regions is the same.

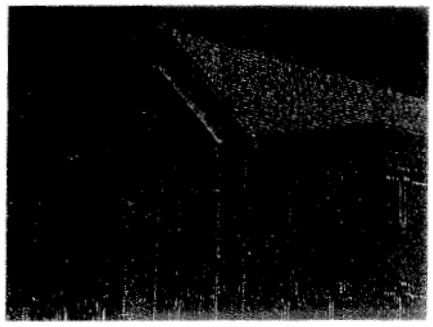


FIGURE 10.17
Gradient angle image computed using Eq. (10.2-11). Areas of constant intensity in this image indicate that the direction of the gradient vector is the same at all the pixel locations in those regions.

As we show in Section 10.2.6, angle information plays a key supporting role in the implementation of the Canny edge detection algorithm, the most advanced edge detection method we discuss in this chapter.

The original image in Fig. 10.16(a) is of reasonably high resolution (834 × 1114 pixels), and at the distance the image was acquired, the contribution made to image detail by the wall bricks is significant. This level of fine detail often is undesirable in edge detection because it tends to act as noise, which is enhanced by derivative computations and thus complicates detection of the principal edges in an image. One way to reduce fine detail is to smooth the image. Figure 10.18 shows the same sequence of images as in Fig. 10.16, but with the original image smoothed first using a 5 × 5 averaging filter (see Section 3.5 regarding smoothing filters). The response of each mask now shows almost no contribution due to the bricks, with the results being dominated mostly by the principal edges.

The maximum edge strength (magnitude) of a smoothed image decreases inversely as a function of the size of the smoothing mask (Problem 10.13).

It is evident in Figs. 10.16 and 10.18 that the horizontal and vertical Sobel masks do not differentiate between edges oriented in the ±45° directions. If it is important to emphasize edges along the diagonal directions, then one of the masks in Fig. 10.15 should be used. Figures 10.19(a) and (b) show the absolute responses of the 45° and −45° Sobel masks, respectively. The stronger diagonal response of these masks is evident in these figures. Both diagonal masks have similar response to horizontal and vertical edges but, as expected, their response in these directions is weaker than the response of the horizontal and vertical masks, as discussed earlier.

a b
c d

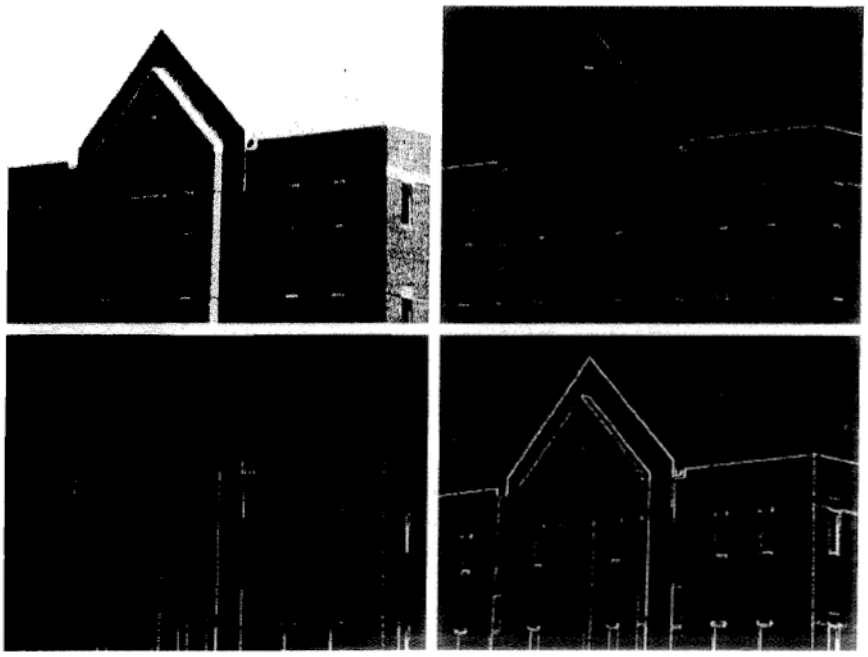


FIGURE 10.18
Same sequence as in Fig. 10.16, but with the original image smoothed using a 5 × 5 averaging filter prior to edge detection.

a b

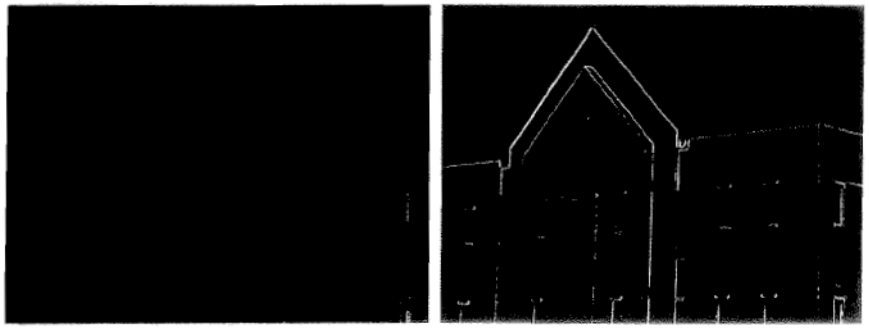
FIGURE 10.19 Diagonal edge detection. (a) Result of using the mask in Fig. 10.15(c). (b) Result of using the mask in Fig. 10.15(d). The input image in both cases was Fig. 10.18(a).

Combining the gradient with thresholding

The results in Fig. 10.18 show that edge detection can be made more selective by smoothing the image prior to computing the gradient. Another approach aimed at achieving the same basic objective is to threshold the gradient image. For example, Fig. 10.20(a) shows the gradient image from Fig. 10.16(d) thresholded, in the sense that pixels with values greater than or equal to 33% of the maximum value of the gradient image are shown in white, while pixels below the threshold value are shown in black. Comparing this image with Fig. 10.18(d), we see that there are fewer edges in the thresholded image, and that the edges in this image are much sharper (see, for example, the edges in the roof tile). On the other hand, numerous edges, such as the 45° line defining the far edge of the roof, are broken in the thresholded image.

The threshold used to generate Fig. 10.20(a) was selected so that most of the small edges caused by the bricks were eliminated. Recall that this was the original objective for smoothing the image in Fig. 10.16 prior to computing the gradient.

When interest lies both in highlighting the principal edges and on maintaining as much connectivity as possible, it is common practice to use both smoothing and thresholding. Figure 10.20(b) shows the result of thresholding Fig. 10.18(d), which is the gradient of the smoothed image. This result shows a



a b

FIGURE 10.20 (a) Thresholded version of the image in Fig. 10.16(d), with the threshold selected as 33% of the highest value in the image; this threshold was just high enough to eliminate most of the brick edges in the gradient image. (b) Thresholded version of the image in Fig. 10.18(d), obtained using a threshold equal to 33% of the highest value in that image.

reduced number of broken edges; for instance, compare the 45° edges in Figs. 10.20(a) and (b). Of course, edges whose intensity values were severely attenuated due to blurring (e.g., the edges in the tile roof) are likely to be totally eliminated by thresholding. We return to the problem of broken edges in Section 10.2.7.

10.2.6 More Advanced Techniques for Edge Detection

The edge-detection methods discussed in the previous section are based simply on filtering an image with one or more masks, with no provisions being made for edge characteristics and noise content. In this section, we discuss more advanced techniques that make an attempt to improve on simple edge-detection methods by taking into account factors such as image noise and the nature of edges themselves.

The Marr-Hildreth edge detector

One of the earliest successful attempts at incorporating more sophisticated analysis into the edge-finding process is attributed to Marr and Hildreth [1980]. Edge-detection methods in use at the time were based on using small operators (such as the Sobel masks), as discussed in the previous section. Marr and Hildreth argued (1) that intensity changes are not independent of image scale and so their detection requires the use of operators of different sizes; and (2) that a sudden intensity change will give rise to a peak or trough in the first derivative or, equivalently, to a zero crossing in the second derivative (as we saw in Fig. 10.10).

To convince yourself that edge detection is not independent of scale, consider, for example, the roof edge in Fig. 10.8(c). If the scale of the image is reduced, the edge will appear thinner.

These ideas suggest that an operator used for edge detection should have two salient features. First and foremost, it should be a differential operator capable of computing a digital approximation of the first or second derivative at every point in the image. Second, it should be capable of being "tuned" to act at any desired scale, so that large operators can be used to detect blurry edges and small operators to detect sharply focused fine detail.

Marr and Hildreth argued that the most satisfactory operator fulfilling these conditions is the filter $\nabla^2 G$ where, as defined in Section 3.6.2, ∇^2 is the Laplacian operator, $(\partial^2/\partial x^2 + \partial^2/\partial y^2)$, and G is the 2-D Gaussian function

$$G(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{10.2-21}$$

It is customary for Eq. (10.2-21) to differ from the definition of a 2-D Gaussian PDF by the constant term $1/2\pi\sigma^2$. If an exact expression is desired in a given application, then the multiplying constant can be appended to the final result in Eq. (10.2-23).

with standard deviation σ (sometimes σ is called the *space constant*). To find an expression for $\nabla^2 G$ we perform the following differentiations:

$$\begin{aligned}\nabla^2 G(x, y) &= \frac{\partial^2 G(x, y)}{\partial x^2} + \frac{\partial^2 G(x, y)}{\partial y^2} \\ &= \frac{\partial}{\partial x}\left[\frac{-x}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}\right] + \frac{\partial}{\partial y}\left[\frac{-y}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}\right] \\ &= \left[\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}} + \left[\frac{y^2}{\sigma^4} - \frac{1}{\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}}\end{aligned} \tag{10.2-22}$$

Collecting terms gives the final expression:

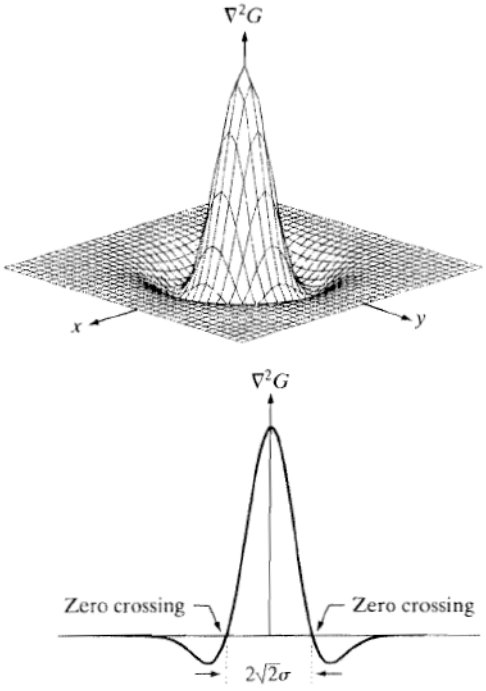
$$\nabla^2 G(x, y) = \left[\frac{x^2 + y^2 - 2\sigma^2}{\sigma^4}\right] e^{-\frac{x^2+y^2}{2\sigma^2}} \qquad (10.2\text{-}23)$$

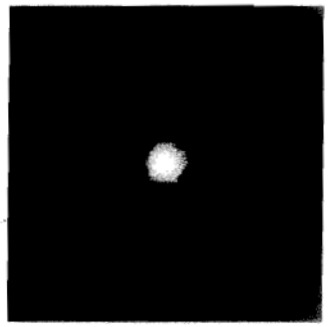
This expression is called the *Laplacian of a Gaussian* (LoG).

Figures 10.21(a) through (c) show a 3-D plot, image, and cross section of the *negative* of the LoG function (note that the zero crossings of the LoG occur at $x^2 + y^2 = 2\sigma^2$, which defines a circle of radius $\sqrt{2}\sigma$ centered on the origin). Because of the shape illustrated in Fig. 10.21(a), the LoG function sometimes is called the *Mexican hat* operator. Figure 10.21(d) shows a 5×5 mask that approximates the shape in Fig. 10.21(a) (in practice we would use the *negative* of this mask). This approximation is not unique. Its purpose is to capture the essential *shape* of the LoG function; in terms of Fig. 10.21(a), this means a positive, central term surrounded by an adjacent, negative region whose values increase as a function of distance from the origin, and a zero outer region. The coefficients must sum to zero so that the response of the mask is zero in areas of constant intensity.

Note the similarity between the cross section in Fig. 10.21(c) and the highpass filter in Fig. 4.37(d). Thus, we can expect the LoG to behave as a highpass filter.

Masks of arbitrary size can be generated by sampling Eq. (10.2-23) and scaling the coefficients so that they sum to zero. A more effective approach for generating a LoG filter is to sample Eq. (10.2-21) to the desired $n \times n$ size and





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

a b
c d

FIGURE 10.21 (a) Three-dimensional plot of the *negative* of the LoG. (b) Negative of the LoG displayed as an image. (c) Cross section of (a) showing zero crossings. (d) 5×5 mask approximation to the shape in (a). The negative of this mask would be used in practice.

then convolve† the resulting array with a Laplacian mask, such as the mask in Fig. 10.4(a). Because convolving an image array with a mask whose coefficients sum to zero yields a result whose elements also sum to zero (see Problems 3.16 and 10.14), this approach automatically satisfies the requirement that the sum of the LoG filter coefficients be zero. We discuss the issue of selecting the size of LoG filter later in this section.

There are two fundamental ideas behind the selection of the operator $\nabla^2 G$. First, the Gaussian part of the operator blurs the image, thus reducing the intensity of structures (including noise) at scales much smaller than σ. Unlike averaging of the form discussed in Section 3.5 and used in Fig. 10.18, the Gaussian function is smooth in both the spatial and frequency domains (see Section 4.8.3), and is thus less likely to introduce artifacts (e.g., ringing) not present in the original image. The other idea concerns ∇^2, the second derivative part of the filter. Although first derivatives can be used for detecting abrupt changes in intensity, they are directional operators. The Laplacian, on the other hand, has the important advantage of being isotropic (invariant to rotation), which not only corresponds to characteristics of the human visual system (Marr [1982]) but also responds equally to changes in intensity in any mask direction, thus avoiding having to use multiple masks to calculate the strongest response at any point in the image.

The Marr-Hildreth algorithm consists of convolving the LoG filter with an input image, $f(x, y)$,

$$g(x, y) = [\nabla^2 G(x, y)] \star f(x, y) \tag{10.2-24}$$

and then finding the zero crossings of $g(x, y)$ to determine the locations of edges in $f(x, y)$. Because these are linear processes, Eq. (10.2-24) can be written also as

$$g(x, y) = \nabla^2 [G(x, y) \star f(x, y)] \tag{10.2-25}$$

This expression is implemented in the spatial domain using Eq. (3.4-2). It can be implemented also in the frequency domain using Eq. (4.7-1).

indicating that we can smooth the image first with a Gaussian filter and then compute the Laplacian of the result. These two equations give identical results.

The Marr-Hildreth edge-detection algorithm may be summarized as follows:

1. Filter the input image with an $n \times n$ Gaussian lowpass filter obtained by sampling Eq. (10.2-21).
2. Compute the Laplacian of the image resulting from Step 1 using, for example, the 3×3 mask in Fig. 10.4(a). [Steps 1 and 2 implement Eq. (10.2-25).]
3. Find the zero crossings of the image from Step 2.

To specify the size of the Gaussian filter, recall that about 99.7% of the volume under a 2-D Gaussian surface lies between $\pm 3\sigma$ about the mean. Thus, as a rule

†The LoG is a symmetric filter, so spatial filtering using correlation or convolution yields the same result. We use the convolution terminology here to indicate linear filtering for consistency with the literature on this topic. Also, this gives you exposure to terminology that you will encounter in other contexts. It is important that you keep in mind the comments made at the end of Section 3.4.2 regarding this topic.

of thumb, the size of an $n \times n$ LoG discrete filter should be such that n is the smallest odd integer greater than or equal to 6σ. Choosing a filter mask smaller than this will tend to "truncate" the LoG function, with the degree of truncation being inversely proportional to the size of the mask; using a larger mask would make little difference in the result.

One approach for finding the zero crossings at any pixel, p, of the filtered image, $g(x, y)$, is based on using a 3×3 neighborhood centered at p. A zero crossing at p implies that the *signs* of at least two of its opposing neighboring pixels must differ. There are four cases to test: left/right, up/down, and the two diagonals. If the values of $g(x, y)$ are being compared against a threshold (a common approach), then not only must the signs of opposing neighbors be different, but the absolute value of their numerical difference must also exceed the threshold before we can call p a zero-crossing pixel. We illustrate this approach in Example 10.7 below.

Attempting to find the zero crossings by finding the coordinates (x, y), such that $g(x, y) = 0$ is impractical because of noise and/or computational inaccuracies.

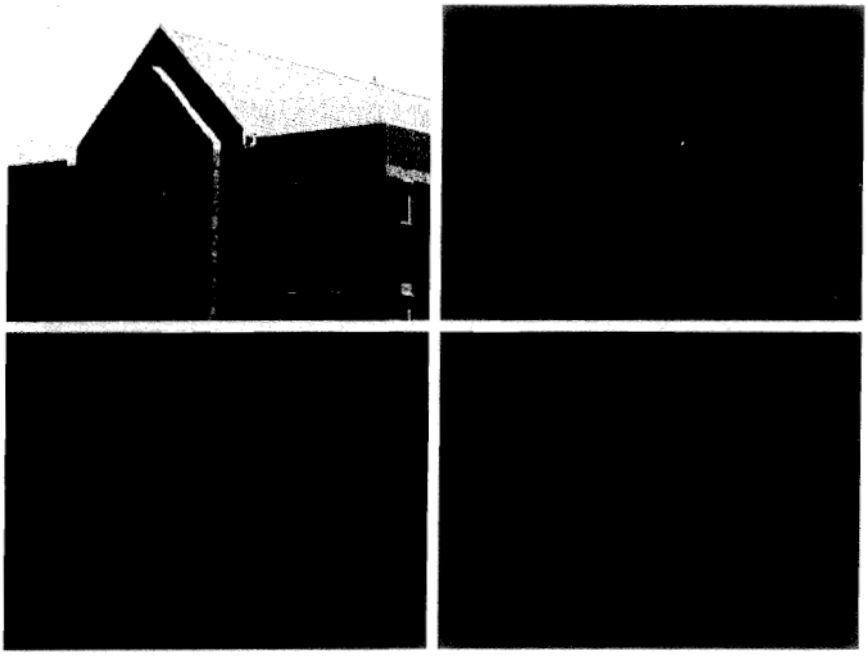
Zero crossings are the key feature of the Marr-Hildreth edge-detection method. The approach discussed in the previous paragraph is attractive because of its simplicity of implementation and because it generally gives good results. If the accuracy of the zero-crossing locations found using this method is inadequate in a particular application, then a technique proposed by Huertas and Medioni [1986] for finding zero crossings with subpixel accuracy can be employed.

EXAMPLE 10.7: Illustration of the Marr-Hildreth edge-detection method.

■ Figure 10.22(a) shows the original building image used earlier and Fig. 10.22(b) is the result of Steps 1 and 2 of the Marr-Hildreth algorithm, using $\sigma = 4$ (approximately 0.5% of the short dimension of the image) and $n = 25$ (the smallest odd integer greater than or equal to 6σ, as discussed earlier). As in Fig. 10.5, the gray tones in this image are due to scaling. Figure 10.22(c) shows the zero crossings obtained using the 3×3 neighborhood approach discussed above with a threshold of zero. Note that all the edges form closed loops. This so-called "spaghetti" effect is a serious drawback of this method when a threshold value of zero is used (Problem 10.15). We avoid closed-loop edges by using a positive threshold.

Figure 10.22(d) shows the result of using a threshold approximately equal to 4% of the maximum value of the LoG image. Note that the majority of the principal edges were readily detected and "irrelevant" features, such as the edges due to the bricks and the tile roof, were filtered out. As we show in the next section, this type of performance is virtually impossible to obtain using the gradient-based edge-detection techniques discussed in the previous section. Another important consequence of using zero crossings for edge detection is that the resulting edges are 1 pixel thick. This property simplifies subsequent stages of processing, such as edge linking. ■

A procedure used sometimes to take into account the fact mentioned earlier that intensity changes are scale dependent is to filter an image with various values of σ. The resulting zero-crossings edge maps are then combined by keeping only the edges that are common to all maps. This approach can yield



a b
c d

FIGURE 10.22
(a) Original image of size 834 × 1114 pixels, with intensity values scaled to the range [0, 1]. (b) Results of Steps 1 and 2 of the Marr-Hildreth algorithm using $\sigma = 4$ and $n = 25$. (c) Zero crossings of (b) using a threshold of 0 (note the closed-loop edges). (d) Zero crossings found using a threshold equal to 4% of the maximum value of the image in (b). Note the thin edges.

useful information, but, due to its complexity, it is used in practice mostly as a design tool for selecting an appropriate value of σ to use with a single filter.

Marr and Hildreth [1980] noted that it is possible to approximate the LoG filter in Eq. (10.2-23) by a difference of Gaussians (DoG):

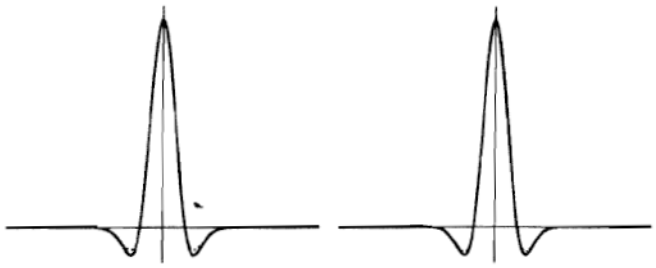
The difference of Gaussians is a highpass filter, as discussed in Section 4.7.4.

$$\mathrm{DoG}(x, y) = \frac{1}{2\pi\sigma_1^2} e^{-\frac{x^2+y^2}{2\sigma_1^2}} - \frac{1}{2\pi\sigma_2^2} e^{-\frac{x^2+y^2}{2\sigma_2^2}} \tag{10.2-26}$$

with $\sigma_1 > \sigma_2$. Experimental results suggest that certain "channels" in the human vision system are selective with respect to orientation and frequency, and can be modeled using Eq. (10.2-26) with a ratio of standard deviations of 1.75:1. Marr and Hildreth suggested that using the ratio 1.6:1 preserves the basic characteristics of these observations and also provides a closer "engineering" approximation to the LoG function. To make meaningful comparisons between the LoG and DoG, the value of σ for the LoG must be selected as in the following equation so that the LoG and DoG have the same zero crossings (Problem 10.17):

$$\sigma^2 = \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2 - \sigma_2^2} \ln\left[\frac{\sigma_1^2}{\sigma_2^2}\right] \tag{10.2-27}$$

Although the zero crossings of the LoG and DoG will be the same when this value of σ is used, their amplitude scales will be different. We can make them compatible by scaling both functions so that they have the same value at the origin.



a b

FIGURE 10.23
(a) Negatives of the LoG (solid) and DoG (dotted) profiles using a standard deviation ratio of 1.75:1. (b) Profiles obtained using a ratio of 1.6:1.

The profiles in Figs. 10.23(a) and (b) were generated with standard deviation ratios of 1:1.75 and 1:1.6, respectively (by convention, the curves shown are inverted, as in Fig. 10.21). The LoG profiles are shown as solid lines while the DoG profiles are dotted. The curves shown are intensity profiles through the center of LoG and DoG arrays generated by sampling Eq. (10.2-23) (with the constant in $1/2\pi\sigma^2$ in front) and Eq. (10.2-26), respectively. The amplitude of all curves at the origin were normalized to 1. As Fig. 10.23(b) shows, the ratio 1:1.6 yielded a closer approximation between the LoG and DoG functions.

Both the LoG and the DoG filtering operations can be implemented with 1-D convolutions instead of using 2-D convolutions directly (Problem 10.19). For an image of size $M \times N$ and a filter of size $n \times n$, doing so reduces the number of multiplications and additions for each convolution from being proportional to n^2MN for 2-D convolutions to being proportional to nMN for 1-D convolutions. This implementation difference is significant. For example, if $n = 25$, a 1-D implementation will require on the order of 12 times fewer multiplication and addition operations than using 2-D convolution.

The Canny edge detector

Although the algorithm is more complex, the performance of the Canny edge detector (Canny [1986]) discussed in this section is superior in general to the edge detectors discussed thus far. Canny's approach is based on three basic objectives:

1. *Low error rate.* All edges should be found, and there should be no spurious responses. That is, the edges detected must be as close as possible to the true edges.
2. *Edge points should be well localized.* The edges located must be as close as possible to the true edges. That is, the distance between a point marked as an edge by the detector and the center of the true edge should be minimum.
3. *Single edge point response.* The detector should return only one point for each true edge point. That is, the number of local maxima around the true edge should be minimum. This means that the detector should not identify multiple edge pixels where only a single edge point exists.

The essence of Canny's work was in expressing the preceding three criteria mathematically and then attempting to find optimal solutions to these formulations. In general, it is difficult (or impossible) to find a closed-form solution

Recall that *white noise* is noise having a frequency spectrum that is continuous and uniform over a specified frequency band. White Gaussian noise is white noise in which the distribution of amplitude values is Gaussian. Gaussian white noise is a good approximation of many real-world situations and generates mathematically tractable models. It has the useful property that its values are statistically independent.

that satisfies all the preceding objectives. However, using numerical optimization with 1-D step edges corrupted by additive white Gaussian noise led to the conclusion that a good approximation† to the optimal step edge detector is the *first derivative of a Gaussian:*

$$\frac{d}{dx}e^{-\frac{x^2}{2\sigma^2}} = \frac{-x}{\sigma^2}e^{-\frac{x^2}{2\sigma^2}} \tag{10.2-28}$$

Generalizing this result to 2-D involves recognizing that the 1-D approach *still applies* in the direction of the edge normal (see Fig. 10.12). Because the direction of the normal is unknown beforehand, this would require applying the 1-D edge detector in all possible directions. This task can be approximated by first smoothing the image with a *circular* 2-D Gaussian function, computing the gradient of the result, and then using the gradient magnitude and direction to estimate edge strength and direction at every point.

Let $f(x, y)$ denote the input image and $G(x, y)$ denote the Gaussian function:

$$G(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{10.2-29}$$

We form a smoothed image, $f_s(x, y)$, by convolving G and f:

$$f_s(x, y) = G(x, y) \star f(x, y) \tag{10.2-30}$$

This operation is followed by computing the gradient magnitude and direction (angle), as discussed in Section 10.2.5:

$$M(x, y) = \sqrt{g_x^2 + g_y^2} \tag{10.2-31}$$

and

$$\alpha(x, y) = \tan^{-1}\left[\frac{g_y}{g_x}\right] \tag{10.2-32}$$

with $g_x = \partial f_s/\partial x$ and $g_y = \partial f_s/\partial y$. Any of the filter mask pairs in Fig. 10.14 can be used to obtain g_x and g_y. Equation (10.2-30) is implemented using an $n \times n$ Gaussian mask whose size is discussed below. Keep in mind that $M(x, y)$ and $\alpha(x, y)$ are arrays of the same size as the image from which they are computed.

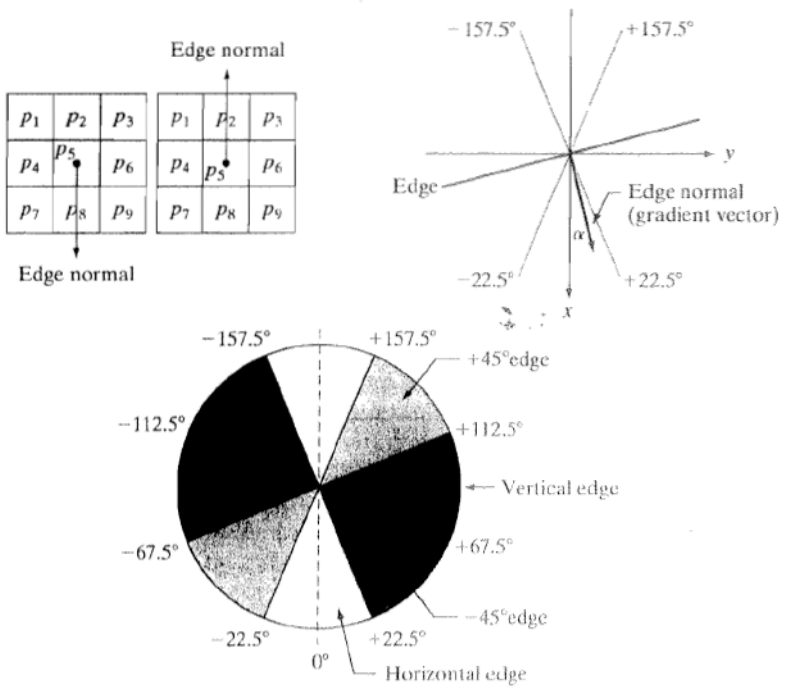
Because it is generated using the gradient, $M(x, y)$ typically contains wide ridges around local maxima (recall the discussion in Section 10.2.1 regarding edges obtained using the gradient). The next step is to thin those ridges. One approach is to use *nonmaxima suppression*. This can be done in several ways, but the essence of the approach is to specify a number of discrete orientations

†Canny [1986] showed that using a Gaussian approximation proved only about 20% worse than using the optimized numerical solution. A difference of this magnitude generally is imperceptible in most applications.

of the edge normal (gradient vector). For example, in a 3×3 region we can define four orientations† for an edge passing through the center point of the region: horizontal, vertical, $+45°$ and $-45°$. Figure 10.24(a) shows the situation for the two possible orientations of a horizontal edge. Because we have to quantize all possible edge directions into four, we have to define a range of directions over which we consider an edge to be horizontal. We determine edge direction from the direction of the edge normal, which we obtain directly from the image data using Eq. (10.2-32). As Fig. 10.24(b) shows, if the edge normal is in the range of directions from $-22.5°$ to $22.5°$ or from $-157.5°$ to $157.5°$, we call the edge a horizontal edge. Figure 10.24(c) shows the angle ranges corresponding to the four directions under consideration.

Let d_1, d_2, d_3, and d_4 denote the four basic edge directions just discussed for a 3×3 region: horizontal, $-45°$, vertical, and $+45°$, respectively. We can formulate the following nonmaxima suppression scheme for a 3×3 region centered at *every* point (x, y) in $\alpha(x, y)$:

1. Find the direction d_k that is closest to $\alpha(x, y)$.
2. If the value of $M(x, y)$ is less than at least one of its two neighbors along d_k, let $g_N(x, y) = 0$ (suppression); otherwise, let $g_N(x, y) = M(x, y)$



a b
c

FIGURE 10.24
(a) Two possible orientations of a horizontal edge (in gray) in a 3×3 neighborhood. (b) Range of values (in gray) of α, the direction angle of the *edge normal*, for a horizontal edge. (c) The angle ranges of the edge normals for the four types of edge directions in a 3×3 neighborhood. Each edge direction has two ranges, shown in corresponding shades of gray.

†Keep in mind that every edge has two possible orientations. For example, an edge whose normal is oriented at 0° and an edge whose normal is oriented at 180° are the same *horizontal* edge.

where $g_N(x, y)$ is the nonmaxima-suppressed image. For example, with reference to Fig. 10.24(a), letting (x, y) be at p_5 and assuming a horizontal edge through p_5, the pixels in which we would be interested in Step 2 are p_2 and p_8. Image $g_N(x, y)$ contains only the thinned edges; it is equal to $M(x, y)$ with the nonmaxima edge points suppressed.

The final operation is to threshold $g_N(x, y)$ to reduce false edge points. In Section 10.2.5 we did this using a single threshold, in which all values below the threshold were set to 0. If we set the threshold too low, there will still be some false edges (called *false positives*). If the threshold is set too high, then actual valid edge points will be eliminated (*false negatives*). Canny's algorithm attempts to improve on this situation by using *hysteresis thresholding* which, as we discuss in Section 10.3.6, uses two thresholds: a low threshold, T_L, and a high threshold, T_H. Canny suggested that the ratio of the high to low threshold should be two or three to one.

We can visualize the thresholding operation as creating two additional images

$$g_{NH}(x, y) = g_N(x, y) \geq T_H \tag{10.2-33}$$

and

$$g_{NL}(x, y) = g_N(x, y) \geq T_L \tag{10.2-34}$$

where, initially, both $g_{NH}(x, y)$ and $g_{NL}(x, y)$ are set to 0. After thresholding, $g_{NH}(x, y)$ will have fewer nonzero pixels than $g_{NL}(x, y)$ in general, but all the nonzero pixels in $g_{NH}(x, y)$ will be contained in $g_{NL}(x, y)$ because the latter image is formed with a lower threshold. We eliminate from $g_{NL}(x, y)$ all the nonzero pixels from $g_{NH}(x, y)$ by letting

$$g_{NL}(x, y) = g_{NL}(x, y) - g_{NH}(x, y) \tag{10.2-35}$$

The nonzero pixels in $g_{NH}(x, y)$ and $g_{NL}(x, y)$ may be viewed as being "strong" and "weak" edge pixels, respectively.

After the thresholding operations, all strong pixels in $g_{NH}(x, y)$ are assumed to be valid edge pixels and are so marked immediately. Depending on the value of T_H, the edges in $g_{NH}(x, y)$ typically have gaps. Longer edges are formed using the following procedure:

(a) Locate the next unvisited edge pixel, p, in $g_{NH}(x, y)$.
(b) Mark as valid edge pixels all the weak pixels in $g_{NL}(x, y)$ that are connected to p using, say, 8-connectivity.
(c) If all nonzero pixels in $g_{NH}(x, y)$ have been visited go to Step d. Else, return to Step a.
(d) Set to zero all pixels in $g_{NL}(x, y)$ that were not marked as valid edge pixels.

At the end of this procedure, the final image output by the Canny algorithm is formed by appending to $g_{NH}(x, y)$ all the nonzero pixels from $g_{NL}(x, y)$.

We used two additional images, $g_{NH}(x, y)$ and $g_{NL}(x, y)$, to simplify the discussion. In practice, hysteresis thresholding can be implemented directly during nonmaxima suppression, and thresholding can be implemented directly on $g_N(x, y)$ by forming a list of strong pixels and the weak pixels connected to them.

Summarizing, the Canny edge detection algorithm consists of the following basic steps:

1. Smooth the input image with a Gaussian filter.
2. Compute the gradient magnitude and angle images.
3. Apply nonmaxima suppression to the gradient magnitude image.
4. Use double thresholding and connectivity analysis to detect and link edges.

Although the edges after nonmaxima suppression are thinner than raw gradient edges, edges thicker than 1 pixel can still remain. To obtain edges 1 pixel thick, it is typical to follow Step 4 with one pass of an edge-thinning algorithm (see Section 9.5.5).

As mentioned earlier, smoothing is accomplished by convolving the input image with a Gaussian mask whose size, $n \times n$, must be specified. We can use the approach discussed in the previous section in connection with the Marr-Hildreth algorithm to determine a value of n. That is, a filter mask generated by sampling Eq. (10.2-29) so that n is the smallest odd integer greater than or equal to 6σ provides essentially the "full" smoothing capability of the Gaussian filter. If practical considerations require a smaller filter mask, then the tradeoff is less smoothing for smaller values of n.

Some final comments on implementation: As noted earlier in the discussion of the Marr-Hildreth edge detector, the 2-D Gaussian function in Eq. (10.2-29) is separable into a product of two 1-D Gaussians. Thus, Step 1 of the Canny algorithm can be formulated as 1-D convolutions that operate on the rows (columns) of the image one at a time and then work on the columns (rows) of the result. Furthermore, if we use the approximations in Eqs. (10.2-12) and (10.2-13), we can also implement the gradient computations required for Step 2 as 1-D convolutions (Problem 10.20).

EXAMPLE 10.8: Illustration of the Canny edge-detection method.

■ Figure 10.25(a) shows the familiar building image. For comparison, Figs. 10.25(b) and (c) show, respectively, the results obtained earlier in Fig. 10.20(b) using the thresholded gradient and Fig. 10.22(d) using the Marr-Hildreth detector. Recall that the parameters used in generating those two images were selected to detect the principal edges while attempting to reduce "irrelevant" features, such as the edges due to the bricks and the tile roof.

Figure 10.25(d) shows the result obtained with the Canny algorithm using the parameters $T_L = 0.04$, $T_H = 0.10$ (2.5 times the value of the low threshold), $\sigma = 4$ and a mask of size 25×25, which corresponds to the smallest odd integer greater than 6σ. These parameters were chosen interactively to achieve the objectives stated in the previous paragraph for the gradient and Marr-Hildreth images. Comparing the Canny image with the other two images, we

a b
c d

FIGURE 10.25
(a) Original image of size 834 × 1114 pixels, with intensity values scaled to the range [0, 1].
(b) Thresholded gradient of smoothed image.
(c) Image obtained using the Marr-Hildreth algorithm.
(d) Image obtained using the Canny algorithm. Note the significant improvement of the Canny image compared to the other two.

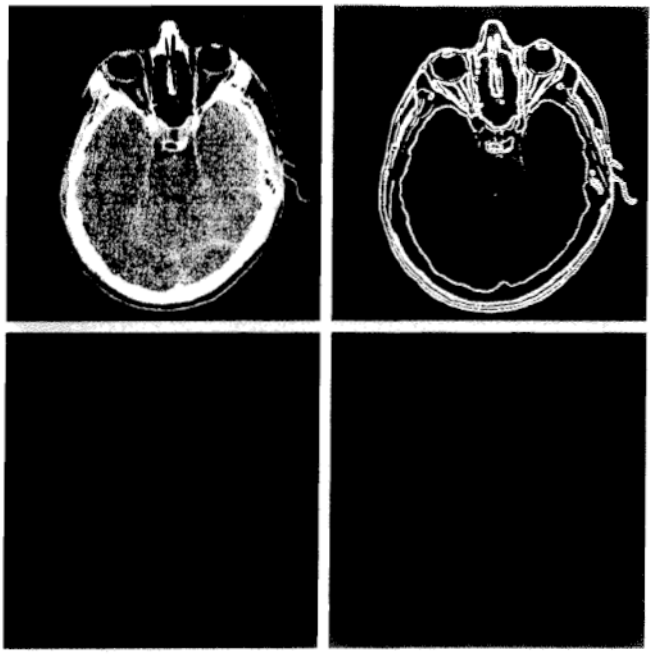
see significant improvements in detail of the principal edges and, at the same time, more rejection of irrelevant features in the Canny result. Note, for example, that both edges of the concrete band lining the bricks in the upper section of the image were detected by the Canny algorithm, whereas the thresholded gradient lost both of these edges and the Marr-Hildreth image contains only the upper one. In terms of filtering out irrelevant detail, the Canny image does not contain a single edge due to the roof tiles; this is not true in the other two images. The quality of the lines with regard to continuity, thinness, and straightness is also superior in the Canny image. Results such as these have made the Canny algorithm a tool of choice for edge detection. ■

The threshold values given here should be considered only in relative terms. Implementation of most algorithms involves various scaling steps, such as scaling the range of values of the input image to the range [0, 1]. Different scaling schemes obviously would require different values of thresholds from those used in this example.

EXAMPLE 10.9: Another illustration of the three principal edge detection methods discussed in this section.

■ As another comparison of the three principal edge-detection methods discussed in this section, consider Fig. 10.26(a) which shows a 512 × 512 head CT image. Our objective in this example is to extract the edges of the outer contour of the brain (the gray region in the image), the contour of the spinal region (shown directly behind the nose, toward the front of the brain), and the outer contour of the head. We wish to generate the thinnest, continuous contours possible, while eliminating edge details related to the gray content in the eyes and brain areas.

Figure 10.26(b) shows a thresholded gradient image that was first smoothed with a 5 × 5 averaging filter. The threshold required to achieve the result shown was 15% of the maximum value of the gradient image. Figure 10.26(c) shows the result obtained with the Marr-Hildreth edge-detection algorithm with a threshold of 0.002, $\sigma = 3$, and a mask of size 19 × 19 pixels. Figure 10.26(d) was obtained using the Canny algorithm with $T_L = 0.05$, $T_H = 0.15$ (3 times the



a b
c d

FIGURE 10.26
(a) Original head CT image of size 512 × 512 pixels, with intensity values scaled to the range [0, 1]. (b) Thresholded gradient of smoothed image. (c) Image obtained using the Marr-Hildreth algorithm. (d) Image obtained using the Canny algorithm. (Original image courtesy of Dr. David R. Pickens, Vanderbilt University.)

value of the low threshold), $\sigma = 2$, and a mask of size 13 × 13, which, as in the Marr-Hildreth case, corresponds to the smallest odd integer greater than 6σ.

The results in Fig. 10.26 correspond closely to the results and conclusions in the previous example in terms of edge quality and the ability to eliminate irrelevant detail. Note also that the Canny algorithm was the only procedure capable of yielding a totally unbroken edge for the posterior boundary of the brain. It was also the only procedure capable of finding the best contours while eliminating all the edges associated with the gray matter in the original image. ■

As might be expected, the price paid for the improved performance of the Canny algorithm is a more complex implementation than the two approaches discussed earlier, requiring also considerably more execution time. In some applications, such as real-time industrial image processing, cost and speed requirements usually dictate the use of simpler techniques, principally the thresholded gradient approach. When edge quality is the driving force, then the Marr-Hildreth and Canny algorithms, especially the latter, offer superior alternatives.

10.2.7 Edge Linking and Boundary Detection

Ideally, edge detection should yield sets of pixels lying only on edges. In practice, these pixels seldom characterize edges completely because of noise, breaks in the edges due to nonuniform illumination, and other effects that introduce spurious discontinuities in intensity values. Therefore, edge detection typically is followed by linking algorithms designed to assemble edge pixels into meaningful edges and/or region boundaries. In this section, we discuss three fundamental approaches to edge linking that are representative of techniques used in practice.

The first requires knowledge about edge points in a local region (e.g., a 3×3 neighborhood); the second requires that points on the boundary of a region be known; and the third is a global approach that works with an entire edge image.

Local processing

One of the simplest approaches for linking edge points is to analyze the characteristics of pixels in a small neighborhood about every point (x, y) that has been declared an edge point by one of the techniques discussed in the previous section. All points that are similar according to predefined criteria are linked, forming an edge of pixels that share common properties according to the specified criteria.

The two principal properties used for establishing similarity of edge pixels in this kind of analysis are (1) the strength (magnitude) and (2) the direction of the gradient vector. The first property is based on Eq. (10.2-10). Let S_{xy} denote the set of coordinates of a neighborhood centered at point (x, y) in an image. An edge pixel with coordinates (s, t) in S_{xy} is similar in *magnitude* to the pixel at (x, y) if

$$|M(s, t) - M(x, y)| \leq E \tag{10.2-36}$$

where E is a positive threshold.

The direction angle of the gradient vector is given by Eq. (10.2-11). An edge pixel with coordinates (s, t) in S_{xy} has an *angle* similar to the pixel at (x, y) if

$$|\alpha(s, t) - \alpha(x, y)| \leq A \tag{10.2-37}$$

where A is a positive angle threshold. As noted in Section 10.2.5, the direction of the edge at (x, y) is *perpendicular* to the direction of the gradient vector at that point.

A pixel with coordinates (s, t) in S_{xy} is linked to the pixel at (x, y) if both magnitude and direction criteria are satisfied. This process is repeated at every location in the image. A record must be kept of linked points as the center of the neighborhood is moved from pixel to pixel. A simple bookkeeping procedure is to assign a different intensity value to each set of linked edge pixels.

The preceding formulation is computationally expensive because all neighbors of every point have to be examined. A simplification particularly well suited for real time applications consists of the following steps:

1. Compute the gradient magnitude and angle arrays, $M(x, y)$ and $\alpha(x, y)$, of the input image, $f(x, y)$.
2. Form a binary image, g, whose value at any pair of coordinates (x, y) is given by:

$$g(x, y) = \begin{cases} 1 & \text{if } M(x, y) > T_M \text{ AND } \alpha(x, y) = A \pm T_A \\ 0 & \text{otherwise} \end{cases}$$

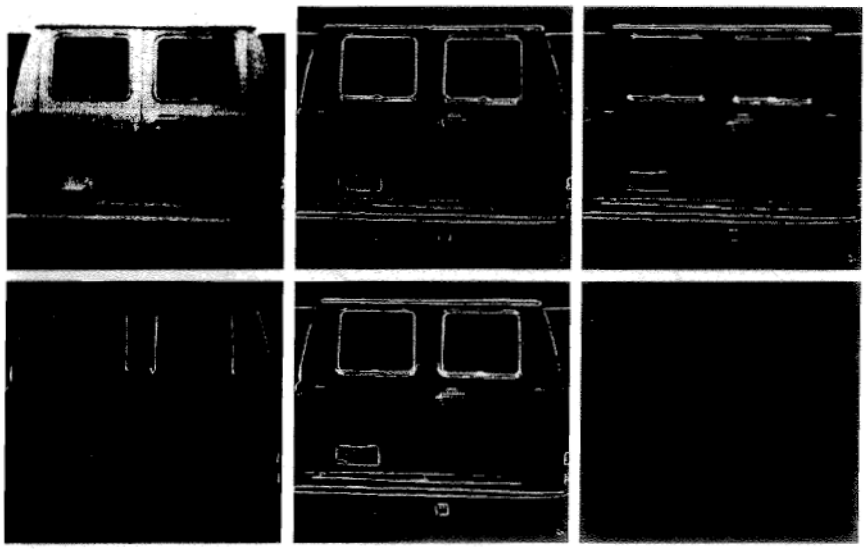
where T_M is a threshold, A is a specified angle direction, and $\pm T_A$ defines a "band" of acceptable directions about A.

3. Scan the rows of g and fill (set to 1) all gaps (sets of 0s) in each row that do not exceed a specified length, K. Note that, by definition, a gap is bounded at both ends by one or more 1s. The rows are processed individually, with no memory between them.
4. To detect gaps in any other direction, θ, rotate g by this angle and apply the horizontal scanning procedure in Step 3. Rotate the result back by $-\theta$.

When interest lies in horizontal and vertical edge linking, Step 4 becomes a simple procedure in which g is rotated ninety degrees, the rows are scanned, and the result is rotated back. This is the application found most frequently in practice and, as the following example shows, this approach can yield good results. In general, image rotation is an expensive computational process so, when linking in numerous angle directions is required, it is more practical to combine Steps 3 and 4 into a single, radial scanning procedure.

EXAMPLE 10.10: Edge linking using local processing.

■ Figure 10.27(a) shows an image of the rear of a vehicle. The objective of this example is to illustrate the use of the preceding algorithm for finding rectangles whose sizes makes them suitable candidates for license plates. The formation of these rectangles can be accomplished by detecting strong horizontal and vertical edges. Figure 10.27(b) shows the gradient magnitude image, $M(x, y)$, and Figs. 10.27(c) and (d) show the result of Steps (3) and (4) of the algorithm obtained by letting T_M equal to 30% of the maximum gradient value,



a b c
d e f

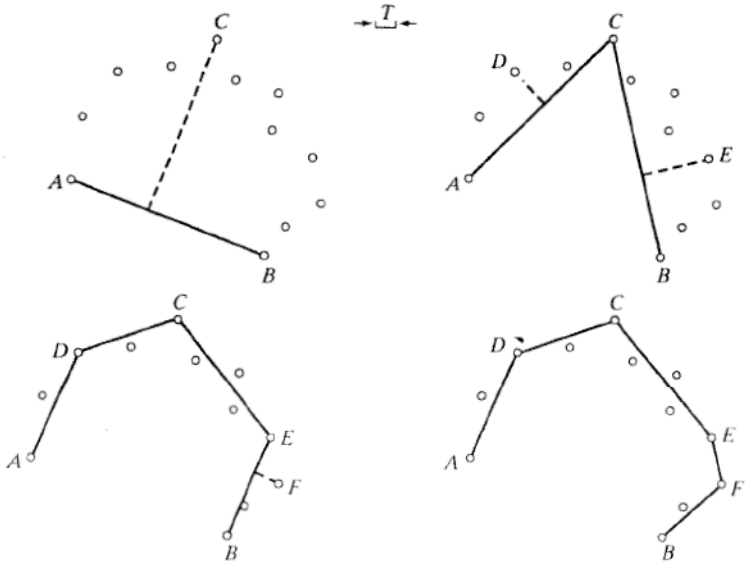
FIGURE 10.27 (a) A 534 × 566 image of the rear of a vehicle. (b) Gradient magnitude image. (c) Horizontally connected edge pixels. (d) Vertically connected edge pixels. (e) The logical OR of the two preceding images. (f) Final result obtained using morphological thinning. (Original image courtesy of Perceptics Corporation.)

$A = 90°$, $T_A = 45°$, and filling in all gaps of 25 or fewer pixels (approximately 5% of the image width). Use of a large range of allowable angle directions was required to detect the rounded corners of the license plate enclosure, as well as the rear windows of the vehicle. Figure 10.27(e) is the result of forming the logical OR of the two preceding images, and Fig. 10.27(f) was obtained by thinning 10.27(e) with the thinning procedure discussed in Section 9.5.5. As Fig. 10.16(f) shows, the rectangle corresponding to the license plate was clearly detected in the image. It would be a simple matter to isolate the license plate from all the rectangles in the image using the fact that the width-to-height ratio of license plates in the U.S. has a distinctive 2:1 proportion. ■

Regional processing

Often, the location of regions of interest in an image are known or can be determined. This implies that knowledge is available regarding the regional membership of pixels in the corresponding edge image. In such situations, we can use techniques for linking pixels on a regional basis, with the desired result being an approximation to the boundary of the region. One approach to this type of processing is functional approximation, where we fit a 2-D curve to the known points. Typically, interest lies in fast-executing techniques that yield an approximation to essential features of the boundary, such as extreme points and concavities. Polygonal approximations are particularly attractive because they can capture the essential shape features of a region while keeping the representation of the boundary (i.e., the vertices of the polygon) relatively simple. In this section, we develop and illustrate an algorithm suitable for this purpose.

Before stating the algorithm, we discuss the mechanics of the procedure using a simple example. Figure 10.28 shows a set of points representing an open curve in which the end points have been labeled A and B. These two



a b
c d

FIGURE 10.28 Illustration of the iterative polygonal fit algorithm.

points are by definition vertices of the polygon. We begin by computing the parameters of a straight line passing through A and B. Then, we compute the perpendicular distance from all other points in the curve to this line and select the point that yielded the maximum distance (ties are resolved arbitrarily). If this distance exceeds a specified threshold, T, the corresponding point, labeled C, is declared a vertex, as Fig. 10.28(a) shows. Lines from A to C and from C to B are then established, and distances from all points between A and C to line AC are obtained. The point corresponding to the maximum distance is declared a vertex, D, if the distance exceeds T; otherwise no new vertices are declared for that segment. A similar procedure is applied to the points between C and B. Figure 10.28(b) shows the result and Fig. 10.28(c) shows the next step. This iterative procedure is continued until no points satisfy the threshold test. Figure 10.28(d) shows the final result which, as you can see, is a reasonable approximation to the shape of a curve fitting the given points.

Two important requirements are implicit in the procedure just explained. First, two starting points must be specified; second, all the points must be ordered (e.g., in a clockwise or counterclockwise direction). When an arbitrary set of points in 2-D does not form a connected path (as is typically the case in edge images) it is not always obvious whether the points belong to a boundary segment (open curve) or a boundary (closed curve). Given that the points are ordered, we can infer whether we are dealing with an open or closed curve by analyzing the distances between points. A large distance between two consecutive points in the ordered sequence relative to the distance between other points as we traverse the sequence of points is a good indication that the curve is open. The end points are then used to start the procedure. If the separation between points tends to be uniform, then we are most likely dealing with a closed curve. In this case, we have several options for selecting the two starting points. One approach is to choose the rightmost and leftmost points in the set. Another is to find the extreme points of the curve (we discuss a way to do this in Section 11.2.1). An algorithm for finding a polygonal fit to open and closed curves may be stated as follows:

1. Let P be a sequence of ordered, distinct, 1-valued points of a binary image. Specify two starting points, A and B. These are the two starting vertices of the polygon.
2. Specify a threshold, T, and two empty stacks, OPEN and CLOSED.
3. If the points in P correspond to a closed curve, put A into OPEN and put B into OPEN *and* into CLOSED. If the points correspond to an open curve, put A into OPEN and B into CLOSED.
4. Compute the parameters of the line passing from the last vertex in CLOSED to the last vertex in OPEN.
5. Compute the distances from the line in Step 4 to all the points in P whose sequence places them between the vertices from Step 4. Select the point, V_{max}, with the maximum distance, D_{max} (ties are resolved arbitrarily).
6. If $D_{max} > T$, place V_{max} at the end of the OPEN stack as a new vertex. Go to Step 4.

See Section 11.1.1 for an algorithm that creates ordered point sequences.

The use of OPEN and CLOSED for the stack names is *not* related to open and closed curves. The stack names indicate simply a stack to store final (CLOSED) vertices or vertices that are in transition (OPEN).

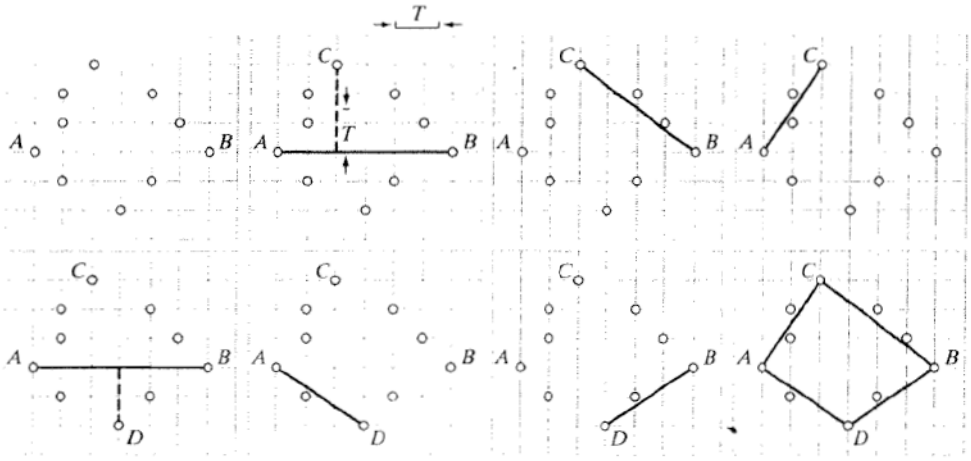
7. Else, remove the last vertex from OPEN and insert it as the last vertex of CLOSED.
8. If OPEN is not empty, go to Step 4.
9. Else, exit. The vertices in CLOSED are the vertices of the polygonal fit to the points in P.

The mechanics of the algorithm are illustrated in the following two examples.

EXAMPLE 10.11: Edge linking using a polygonal approximation.

■ Consider the set of points, P, in Fig. 10.29(a). Assume that these points belong to a closed curve, that they are ordered in a clockwise direction (note that some of the points are not adjacent), and that A and B are selected to be the leftmost and rightmost points in P, respectively. These are the starting vertices, as Table 10.1 shows. Select the first point in the sequence to be the leftmost point, A. Figure 10.29(b) shows the only point (labeled C) in the upper curve segment between A and B that satisfied Step 6 of the algorithm, so it is designated as a new vertex and added to the vertices in the OPEN stack. The second row in Table 10.1 shows C being detected, and the third row shows it being added as the last vertex in OPEN. The threshold, T, in Fig. 10.29(b) is approximately equal to 1.5 subdivisions in the figure grid.

Note in Fig. 10.29(b) that there is a point below line AB that also satisfies Step 6. However, because the points are ordered, only one subset of the points between these two vertices is detected at one time. The other point in the lower segment will be detected later, as Fig. 10.29(e) shows. The key is always to follow the points in the order in which they are given.



a b c d
e f g h

FIGURE 10.29 (a) A set of points in a clockwise path (the points labeled A and B were chosen as the starting vertices). (b) The distance from point C to the line passing through A and B is the largest of all the points between A and B and also passed the threshold test, so C is a new vertex. (d)–(g) Various stages of the algorithm. (h) The final vertices, shown connected with straight lines to form a polygon. Table 10.1 shows step-by-step details.

CLOSED	OPEN	Curve segment processed	Vertex generated
B	B, A	—	A, B
B	B, A	(BA)	C
B	B, A, C	(BC)	—
B, C	B, A	(CA)	—
B, C, A	B	(AB)	D
B, C, A	B, D	(AD)	—
B, C, A, D	B	(DB)	—
B, C, A, D, B	Empty	—	—

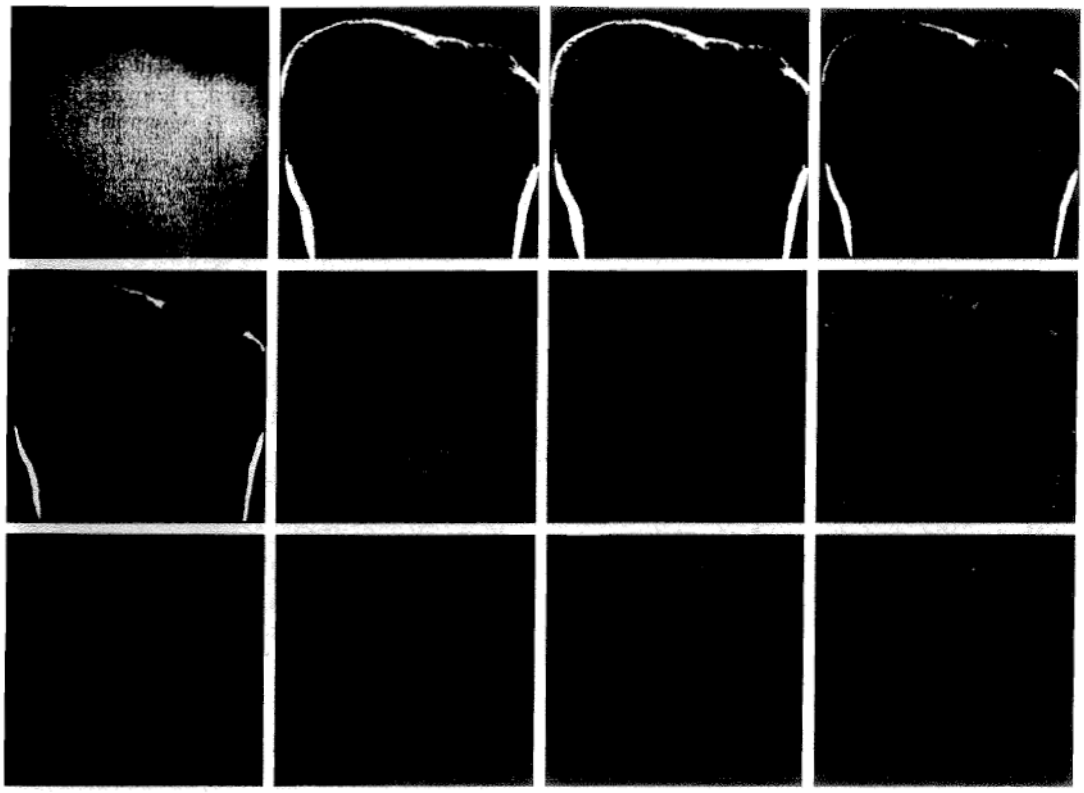
TABLE 10.1 Step-by-step details of the mechanics in Example 10.11.

Table 10.1 shows the individual steps leading to the solution in Fig. 10.29(h). Four vertices were detected, and the figure shows them connected with straight line segments to form a polygon approximating the given boundary points. Note in the table that the vertices detected, B, C, A, D, B are in the counterclockwise direction, even though the points were followed in a clockwise direction to generate the vertices. Had the input been an open curve, the vertices would have been in a clockwise order. The reason for the discrepancy is the way in which the OPEN and CLOSED stacks are initialized. The difference in which stack CLOSED is formed for open and closed curves also leads to the first and last vertices in a closed curve being repeated. This is consistent with how one would differentiate between open and closed polygons given only the vertices. ■

EXAMPLE 10.12: Polygonal fitting of an image boundary.

■ Figure 10.30 shows a more practical example of polygonal fitting. The input image in Fig. 10.30(a) is a 550 × 566 X-ray image of a human tooth with intensities scaled to the interval [0, 1]. The objective of this example is to extract the boundary of the tooth, a process useful in areas such as matching against a database for forensics purposes. Figure 10.30(b) is a gradient image obtained using the Sobel masks and thresholded with $T = 0.1$ (10% of the maximum intensity). As expected for an X-ray image, the noise content is high, so the first step is noise reduction. Because the image is binary, morphological techniques are well suited for this purpose. Figure 10.30(c) shows the result of *majority filtering*, which sets a pixel to 1 if five or more pixels in its 3 × 3 neighborhood are 1 and sets the pixel to 0 otherwise. Although the noise was reduced, some noise points are still clearly visible. Figure 10.30(d) shows the result of morphological shrinking, which further reduced the noise to isolated points. These were eliminated [Fig. 10.30(e)] by morphological filtering in the manner described in Example 9.4. At this point, the image consists of thick boundaries, which can be thinned by obtaining the morphological skeleton, as Fig. 10.30(f) shows. Finally, Fig. 10.30(g) shows the last step in preprocessing using spur reduction, as discussed in Section 9.5.8.

Next, we fit the points in Fig. 10.30(g) with a polygon. Figures 10.30(h)–(j) show the result of using the polygon fitting algorithm with thresholds equal to 0.5%, 1%, and 2% of the image width ($T = 3$, 6, and 12). The first two results are good approximations to the boundary, but the third is marginal. Excessive jaggedness in all three cases clearly indicates that boundary smoothing is



a b c d
e f g h
i j k l

FIGURE 10.30 (a) A 550 × 566 X-ray image of a human tooth. (b) Gradient image. (c) Result of majority filtering. (d) Result of morphological shrinking. (e) Result of morphological cleaning. (f) Skeleton. (g) Spur reduction. (h)–(j) Polygonal fit using thresholds of approximately 0.5%, 1%, and 2% of image width (T = 3, 6, and 12). (k) Boundary in (j) smoothed with a 1-D averaging filter of size 1 × 31 (approximately 5% of image width). (l) Boundary in (h) smoothed with the same filter.

required. Figures 10.30(k) and (l) show the result of convolving a 1-D averaging mask with the boundaries in (j) and (h), respectively. The mask used was a 1 × 31 array of 1s, corresponding approximately to 5% of the image width. As expected, the result in Fig. 10.30(k) again is marginal in terms of preserving important shape features (e.g., the right side is severely distorted). On the other hand, the result in Fig. 10.30(l) shows significant boundary smoothing and reasonable preservation of shape features. For example, the roundness of the left-upper cusp and the details of the right-upper cusp were preserved with reasonable fidelity. ■

The results in the preceding example are typical of what can be achieved with the polygon fitting algorithm discussed in this section. The advantage of this

algorithm is that it is simple to implement and yields results that generally are quite acceptable. In Section 11.1.3, we discuss a more sophisticated procedure capable of yielding closer fits by computing minimum-perimeter polygons.

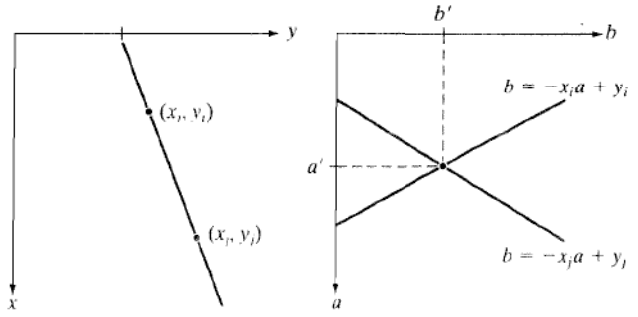
Global processing using the Hough transform

The methods discussed in the previous two sections are applicable in situations where knowledge about pixels belonging to individual objects is at least partially available. For example, in regional processing, it makes sense to link a given set of pixels only if we know that they are part of the boundary of a meaningful region. Often, we have to work with unstructured environments in which all we have is an edge image and no knowledge about where objects of interest might be. In such situations, all pixels are candidates for linking and thus have to be accepted or eliminated based on predefined *global* properties. In this section, we develop an approach based on whether sets of pixels lie on curves of a specified shape. Once detected, these curves form the edges or region boundaries of interest.

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

Hough [1962] proposed an alternative approach, commonly referred to as the *Hough transform*. Consider a point (x_i, y_i) in the xy-plane and the general equation of a straight line in slope-intercept form, $y_i = ax_i + b$. Infinitely many lines pass through (x_i, y_i), but they all satisfy the equation $y_i = ax_i + b$ for varying values of a and b. However, writing this equation as $b = -x_i a + y_i$ and considering the ab-plane (also called *parameter space*) yields the equation of a *single* line for a fixed pair (x_i, y_i). Furthermore, a second point (x_j, y_j) also has a line in parameter space associated with it, and, unless they are parallel, this line intersects the line associated with (x_i, y_i) at some point (a', b'), where a' is the slope and b' the intercept of the line containing *both* (x_i, y_i) and (x_j, y_j) in the xy-plane. In fact, *all* the points on this line have lines in parameter space that intersect at (a', b'). Figure 10.31 illustrates these concepts.

In principle, the parameter-space lines corresponding to all points (x_k, y_k) in the xy-plane could be plotted, and the principal lines in that plane could be found by identifying points in parameter space where large numbers of parameter-space lines intersect. A practical difficulty with this approach, however, is that a



a b

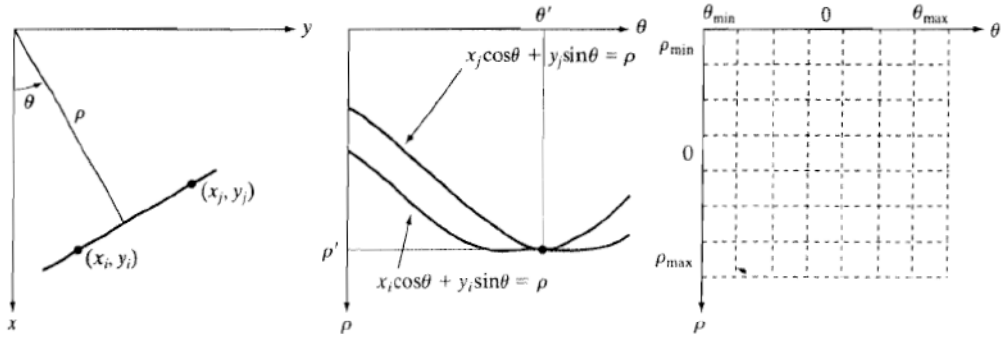
FIGURE 10.31 (a) xy-plane. (b) Parameter space.

(the slope of a line) approaches infinity as the line approaches the vertical direction. One way around this difficulty is to use the normal representation of a line:

$$x \cos\theta + y \sin\theta = \rho \tag{10.2-38}$$

Figure 10.32(a) illustrates the geometrical interpretation of the parameters ρ and θ. A horizontal line has $\theta = 0°$, with ρ being equal to the positive x-intercept. Similarly, a vertical line has $\theta = 90°$, with ρ being equal to the positive y-intercept, or $\theta = -90°$, with ρ being equal to the negative y-intercept. Each sinusoidal curve in Figure 10.32(b) represents the family of lines that pass through a particular point (x_k, y_k) in the xy-plane. The intersection point (ρ', θ') in Fig. 10.32(b) corresponds to the line that passes through both (x_i, y_i) and (x_j, y_j) in Fig. 10.32(a).

The computational attractiveness of the Hough transform arises from subdividing the $\rho\theta$ parameter space into so-called *accumulator cells*, as Fig. 10.32(c) illustrates, where $(\rho_{\min}, \rho_{\max})$ and $(\theta_{\min}, \theta_{\max})$ are the expected ranges of the parameter values: $-90° \le \theta \le 90°$ and $-D \le \rho \le D$, where D is the maximum distance between opposite corners in an image. The cell at coordinates (i, j), with accumulator value $A(i, j)$, corresponds to the square associated with parameter-space coordinates (ρ_i, θ_j). Initially, these cells are set to zero. Then, for every non-background point (x_k, y_k) in the xy-plane, we let θ equal each of the allowed subdivision values on the θ-axis and solve for the corresponding ρ using the equation $\rho = x_k \cos\theta + y_k \sin\theta$. The resulting ρ values are then rounded off to the nearest allowed cell value along the ρ axis. If a choice of θ_p results in solution ρ_q, then we let $A(p, q) = A(p, q) + 1$. At the end of this procedure, a value of P in $A(i, j)$ means that P points in the xy-plane lie on the line $x \cos\theta_j + y \sin\theta_j = \rho_i$. The number of subdivisions in the $\rho\theta$-plane determines the accuracy of the colinearity of these points. It can be shown (Problem 10.24) that the number of computations in the method just discussed is linear with respect to n, the number of non-background points in the xy-plane.



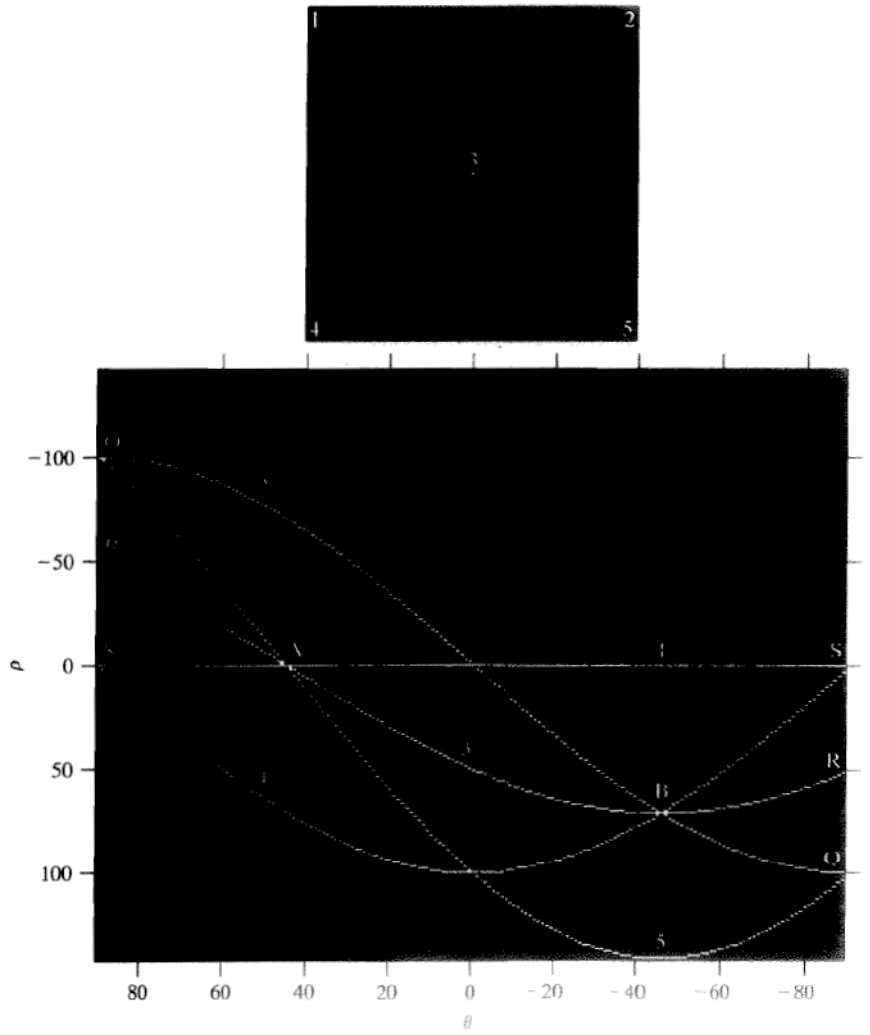
a b c

FIGURE 10.32 (a) (ρ, θ) parameterization of line in the xy-plane. (b) Sinusoidal curves in the $\rho\theta$-plane; the point of intersection (ρ', θ') corresponds to the line passing through points (x_i, y_i) and (x_j, y_j) in the xy-plane. (c) Division of the $\rho\theta$-plane into accumulator cells.

EXAMPLE 10.13: An illustration of basic Hough transform properties.

■ **Figure 10.33** illustrates the Hough transform based on Eq. (10.2-38). **Figure 10.33(a)** shows an image of size 101 × 101 pixels with five labeled points, and **Fig. 10.33(b)** shows each of these points mapped onto the $\rho\theta$-plane using subdivisions of one unit for the ρ and θ axes. The range of θ values is $\pm 90°$, and the range of the ρ axis is $\pm\sqrt{2}D$, where D is the distance between corners in the image. As Fig. 10.33(c) shows, each curve has a different sinusoidal shape. The horizontal line resulting from the mapping of point 1 is a special case of a sinusoid with zero amplitude.

The points labeled A (not to be confused with accumulator values) and B in Fig. 10.33(b) show the colinearity detection property of the Hough transform.



a
b

FIGURE 10.33
(a) Image of size 101 × 101 pixels, containing five points.
(b) Corresponding parameter space. (The points in (a) were enlarged to make them easier to see.)

Point A denotes the intersection of the curves corresponding to points 1, 3, and 5 in the xy image plane. The location of point A indicates that these three points lie on a straight line passing through the origin ($\rho = 0$) and oriented at 45° [see Fig. 10.32(a)]. Similarly, the curves intersecting at point B in the parameter space indicate that points 2, 3, and 4 lie on a straight line oriented at −45°, and whose distance from the origin is $\rho = 71$ (one-half the diagonal distance from the origin of the image to the opposite corner, rounded to the nearest integer value). Finally, the points labeled Q, R, and S in Fig. 10.33(b) illustrate the fact that the Hough transform exhibits a reflective adjacency relationship at the right and left edges of the parameter space. This property is the result of the manner in which θ and ρ change sign at the $\pm 90°$ boundaries. ■

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

$$(x - c_1)^2 + (y - c_2)^2 = c_3^2 \tag{10.2-39}$$

can be detected by using the basic approach just discussed. The difference is the presence of three parameters (c_1, c_2, and c_3), which results in a 3-D parameter space with cube-like cells and accumulators of the form $A(i, j, k)$. The procedure is to increment c_1 and c_2, solve for the c_3 that satisfies Eq. (10.2-39), and update the accumulator cell associated with the triplet (c_1, c_2, c_3). Clearly, the complexity of the Hough transform depends on the number of coordinates and coefficients in a given functional representation. Further generalizations of the Hough transform to detect curves with no simple analytic representations are possible, as is the application of the transform to gray-scale images. Several references dealing with these extensions are included at the end of this chapter.

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

1. Obtain a *binary* edge image using any of the techniques discussed earlier in this section.
2. Specify subdivisions in the $\rho\theta$-plane.
3. Examine the counts of the accumulator cells for high pixel concentrations.
4. Examine the relationship (principally for continuity) between pixels in a chosen cell.

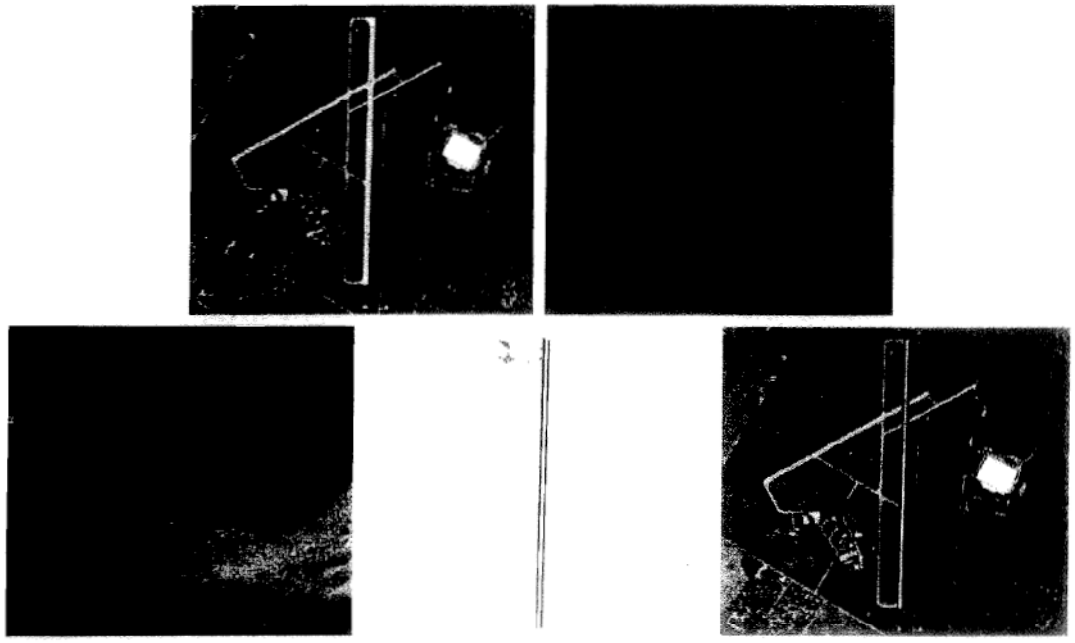
Continuity in this case usually is based on computing the distance between disconnected pixels corresponding to a given accumulator cell. A gap in a line associated with a given cell is bridged if the length of the gap is less than a specified threshold. Note that the mere fact of being able to group lines based on direction is a *global* concept applicable over the entire image, requiring only that we examine pixels associated with specific accumulator cells. This is a significant advantage over the methods discussed in the previous two sections. The following example illustrates these concepts.

EXAMPLE 10.14: Using the Hough transform for edge linking.

■ Figure 10.34(a) shows an aerial image of an airport. The objective of this example is to use the Hough transform to extract the two edges of the principal runway. A solution to such a problem might be of interest, for instance, in applications involving autonomous navigation of air vehicles.

The first step is to obtain an edge image. Figure 10.34(b) shows the edge image obtained using Canny's algorithm with the same parameters and procedure used in Example 10.9. For the purpose of computing the Hough transform, similar results can be obtained using any of the edge-detection techniques discussed in Sections 10.2.5 or 10.2.6. Figure 10.34(c) shows the Hough parameter space obtained using 1° increments for θ and 1 pixel increments for ρ.

The runway of interest is oriented approximately 1° off the north direction, so we select the cells corresponding to $\pm 90°$ and containing the highest count because the runways are the longest lines oriented in these directions. The small white boxes on the edges of Fig. 10.34(c) highlight these cells. As mentioned earlier in connection with Fig. 10.33(b), the Hough transform exhibits adjacency at the edges. Another way of interpreting this property is that a line oriented at $+90°$ and a line oriented at $-90°$ are equivalent (i.e., they are both vertical). Figure 10.34(d) shows the lines corresponding to the two accumulator cells just discussed, and Fig. 10.34(e) shows the lines superimposed on the



a b
c d e

FIGURE 10.34 (a) A 502 × 564 aerial image of an airport. (b) Edge image obtained using Canny's algorithm. (c) Hough parameter space (the boxes highlight the points associated with long vertical lines). (d) Lines in the image plane corresponding to the points highlighted by the boxes). (e) Lines superimposed on the original image.

original image. The lines were obtained by joining all gaps not exceeding 20% of the image height (approximately 100 pixels). These lines clearly correspond to the edges of the runway of interest.

Note that the only key knowledge needed to solve this problem was the orientation of the runway and the observer's position relative to it. In other words, a vehicle navigating autonomously would know that if the runway of interest faces north, and the vehicle's direction of travel also is north, the runway should appear vertically in the image. Other relative orientations are handled in a similar manner. The orientations of runways throughout the world are available in flight charts, and direction of travel is easily obtainable using GPS (Global Positioning System) information. This information also could be used to compute the distance between the vehicle and the runway, thus allowing estimates of parameters such as expected length of lines relative to image size, as we did in this example.

10.3 Thresholding

Because of its intuitive properties, simplicity of implementation, and computational speed, image thresholding enjoys a central position in applications of image segmentation. Thresholding was introduced in Section 3.1.1, and we have used it in various discussions since then. In this section, we discuss thresholding in a more formal way and develop techniques that are considerably more general than what has been presented thus far.

10.3.1 Foundation

In the previous section, regions were identified by first finding edge segments and then attempting to link the segments into boundaries. In this section, we discuss techniques for partitioning images directly into regions based on intensity values and/or properties of these values.

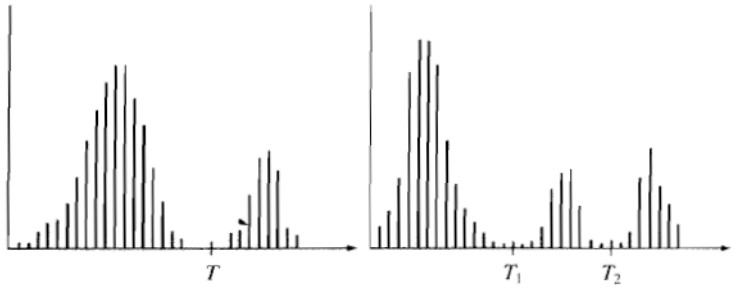
The basics of intensity thresholding

Suppose that the intensity histogram in Fig. 10.35(a) corresponds to an image, $f(x, y)$, composed of light objects on a dark background, in such a way that object and background pixels have intensity values grouped into two dominant modes. One obvious way to extract the objects from the background is to select a threshold, T, that separates these modes. Then, any point (x, y) in the image at which $f(x, y) > T$ is called an *object point*; otherwise, the point is called a *background point*. In other words, the segmented image, $g(x, y)$, is given by

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

Although we follow convention in using 0 intensity for the background and 1 for object pixels, any two distinct values can be used in Eq. (10.3-1).

When T is a constant applicable over an entire image, the process given in this equation is referred to as *global thresholding*. When the value of T changes over an image, we use the term *variable thresholding*. The term *local* or *regional thresholding* is used sometimes to denote variable thresholding in



a b

FIGURE 10.35 Intensity histograms that can be partitioned (a) by a single threshold, and (b) by dual thresholds.

which the value of T at any point (x, y) in an image depends on properties of a neighborhood of (x, y) (for example, the average intensity of the pixels in the neighborhood). If T depends on the spatial coordinates (x, y) themselves, then variable thresholding is often referred to as *dynamic* or *adaptive* thresholding. Use of these terms is not universal, and one is likely to see them used interchangeably in the literature on image processing.

Figure 10.35(b) shows a more difficult thresholding problem involving a histogram with three dominant modes corresponding, for example, to two types of light objects on a dark background. Here, *multiple thresholding* classifies a point (x, y) as belonging to the background if $f(x, y) \leq T_1$, to one object class if $T_1 < f(x, y) \leq T_2$, and to the other object class if $f(x, y) > T_2$. That is, the segmented image is given by

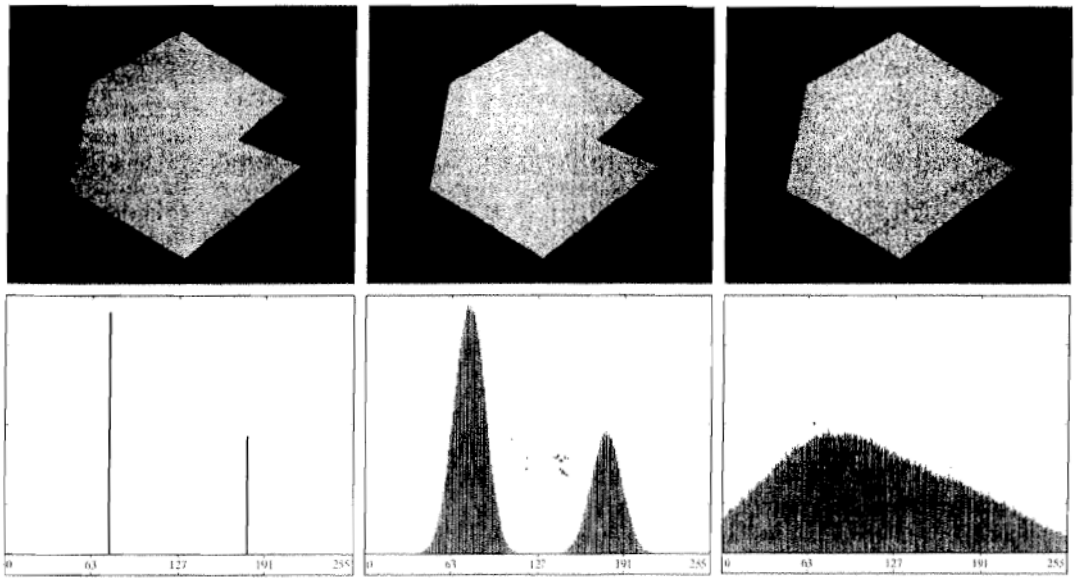
$$g(x, y) = \begin{cases} a & \text{if } f(x, y) > T_2 \\ b & \text{if } T_1 < f(x, y) \leq T_2 \\ c & \text{if } f(x, y) \leq T_1 \end{cases} \tag{10.3-2}$$

where a, b, and c are any three distinct intensity values. We discuss dual thresholding in Section 10.3.6. Segmentation problems requiring more than two thresholds are difficult (often impossible) to solve, and better results usually are obtained using other methods, such as variable thresholding, as discussed in Section 10.3.7, or region growing, as discussed in Section 10.4.

Based on the preceding discussion, we may infer intuitively that the success of intensity thresholding is directly related to the width and depth of the valley(s) separating the histogram modes. In turn, the key factors affecting the properties of the valley(s) are: (1) the separation between peaks (the further apart the peaks are, the better the chances of separating the modes); (2) the noise content in the image (the modes broaden as noise increases); (3) the relative sizes of objects and background; (4) the uniformity of the illumination source; and (5) the uniformity of the reflectance properties of the image.

The role of noise in image thresholding

As an illustration of how noise affects the histogram of an image, consider Fig. 10.36(a). This simple synthetic image is free of noise, so its histogram consists of two "spike" modes, as Fig. 10.36(d) shows. Segmenting this image into two regions is a trivial task involving a threshold placed anywhere between the two



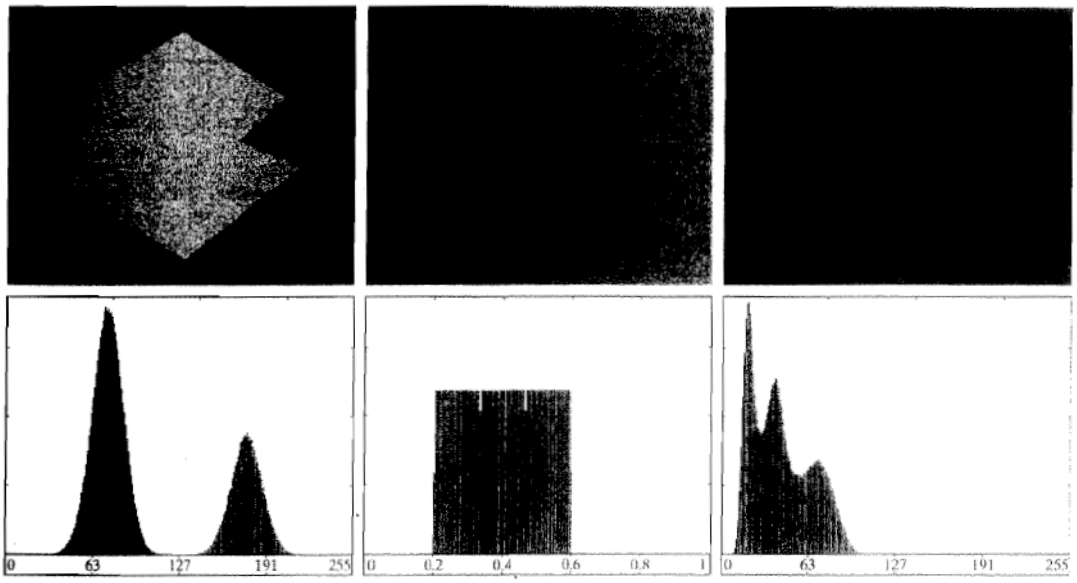
a b c
d e f

FIGURE 10.36 (a) Noiseless 8-bit image. (b) Image with additive Gaussian noise of mean 0 and standard deviation of 10 intensity levels. (c) Image with additive Gaussian noise of mean 0 and standard deviation of 50 intensity levels. (d)–(f) Corresponding histograms.

modes. Figure 10.36(b) shows the original image corrupted by Gaussian noise of zero mean and a standard deviation of 10 intensity levels. Although the corresponding histogram modes are now broader [Fig. 10.36(e)], their separation is large enough so that the depth of the valley between them is sufficient to make the modes easy to separate. A threshold placed midway between the two peaks would do a nice job of segmenting the image. Figure 10.36(c) shows the result of corrupting the image with Gaussian noise of zero mean and a standard deviation of 50 intensity levels. As the histogram in Fig. 10.36(f) shows, the situation is much more serious now, as there is no way to differentiate between the two modes. Without additional processing (such as the methods discussed in Sections 10.3.4 and 10.3.5) we have little hope of finding a suitable threshold for segmenting this image.

The role of illumination and reflectance

Figure 10.37 illustrates the effect that illumination can have on the histogram of an image. Figure 10.37(a) is the noisy image from Fig. 10.36(b), and Fig. 10.37(d) shows its histogram. As before, this image is easily segmentable with a single threshold. We can illustrate the effects of nonuniform illumination by multiplying the image in Fig. 10.37(a) by a variable intensity function, such as the intensity ramp in Fig. 10.37(b), whose histogram is shown in Fig. 10.37(e). Figure 10.37(c) shows the product of the image and this shading pattern. As Fig. 10.37(f) shows, the deep valley between peaks was corrupted to the point



a b c
d e f

FIGURE 10.37 (a) Noisy image. (b) Intensity ramp in the range [0.2, 0.6]. (c) Product of (a) and (b). (d)–(f) Corresponding histograms.

where separation of the modes without additional processing (see Sections 10.3.4 and 10.3.5) is no longer possible. Similar results would be obtained if the illumination was perfectly uniform, but the reflectance of the image was not, due, for example, to natural reflectivity variations in the surface of objects and/or background.

The key point in the preceding paragraph is that illumination and reflectance play a central role in the success of image segmentation using thresholding or other segmentation techniques. Therefore, controlling these factors when it is possible to do so should be the first step considered in the solution of a segmentation problem. There are three basic approaches to the problem when control over these factors is not possible. One is to correct the shading pattern directly. For example, nonuniform (but fixed) illumination can be corrected by multiplying the image by the inverse of the pattern, which can be obtained by imaging a flat surface of constant intensity. The second approach is to attempt to correct the global shading pattern via processing using, for example, the top-hat transformation introduced in Section 9.6.3. The third approach is to "work around" nonuniformities using variable thresholding, as discussed in Section 10.3.7.

In theory, the histogram of a ramp image is uniform. In practice, achieving perfect uniformity depends on the size of the image and number of intensity bits. For example, a 256 × 256, 256-level ramp image has a uniform histogram, but a 256 × 257 ramp image with the same number of intensities does not.

10.3.2 Basic Global Thresholding

As noted in the previous section, when the intensity distributions of objects and background pixels are sufficiently distinct, it is possible to use a single (*global*) threshold applicable over the entire image. In most applications, there

is usually enough variability between images that, even if global thresholding is a suitable approach, an algorithm capable of estimating automatically the threshold value for each image is required. The following iterative algorithm can be used for this purpose:

1. Select an initial estimate for the global threshold, T.
2. Segment the image using T in Eq. (10.3-1). This will produce two groups of pixels: G_1 consisting of all pixels with intensity values $> T$, and G_2 consisting of pixels with values $\leq T$.
3. Compute the average (mean) intensity values m_1 and m_2 for the pixels in G_1 and G_2, respectively.
4. Compute a new threshold value:

$$T = \frac{1}{2}(m_1 + m_2)$$

5. Repeat Steps 2 through 4 until the difference between values of T in successive iterations is smaller than a predefined parameter ΔT.

This simple algorithm works well in situations where there is a reasonably clear valley between the modes of the histogram related to objects and background. Parameter ΔT is used to control the number of iterations in situations where speed is an important issue. In general, the larger ΔT is, the fewer iterations the algorithm will perform. The initial threshold must be chosen greater than the minimum and less than maximum intensity level in the image (Problem 10.28). The average intensity of the image is a good initial choice for T.

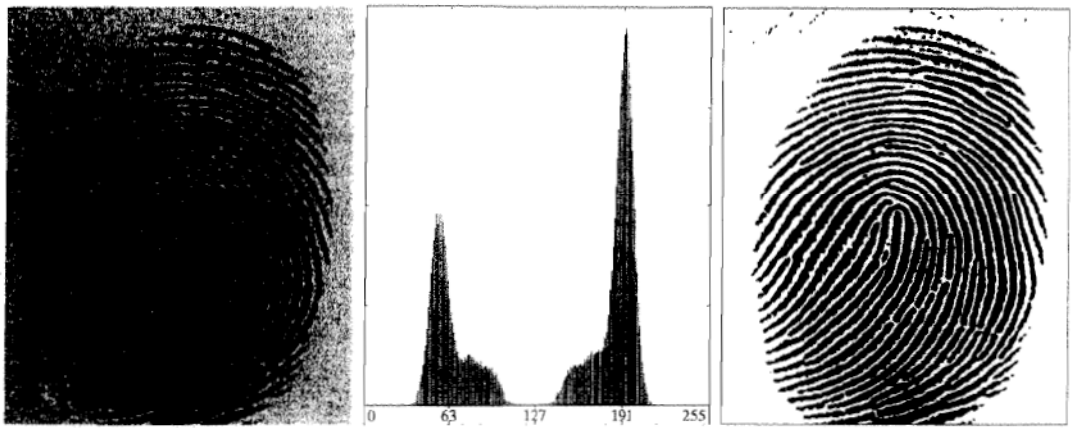
EXAMPLE 10.15: Global thresholding.

■ Figure 10.38 shows an example of segmentation based on a threshold estimated using the preceding algorithm. Figure 10.38(a) is the original image, and Fig. 10.38(b) is the image histogram, showing a distinct valley. Application of the preceding iterative algorithm resulted in the threshold $T = 125.4$ after three iterations, starting with $T = m$ (the average image intensity), and using $\Delta T = 0$. Figure 10.38(c) shows the result obtained using $T = 125$ to segment the original image. As expected from the clear separation of modes in the histogram, the segmentation between object and background was quite effective. ■

The preceding algorithm was stated in terms of successively thresholding the input image and calculating the means at each step because it is more intuitive to introduce it in this manner. However, it is possible to develop a more efficient procedure by expressing all computations in the terms of the image histogram, which has to be computed only once (Problem 10.26).

10.3.3 Optimum Global Thresholding Using Otsu's Method

Thresholding may be viewed as a statistical-decision theory problem whose objective is to minimize the average error incurred in assigning pixels to two or more groups (also called *classes*). This problem is known to have an elegant closed-form solution known as the *Bayes decision rule* (see Section 12.2.2). The solution is based on only two parameters: the probability density function (PDF) of the intensity levels of each class and the probability that each class occurs in a given application. Unfortunately, estimating PDFs is not a trivial



a b c

FIGURE 10.38 (a) Noisy fingerprint. (b) Histogram. (c) Segmented result using a global threshold (the border was added for clarity). (Original courtesy of the National Institute of Standards and Technology.)

matter, so the problem usually is simplified by making workable assumptions about the form of the PDFs, such as assuming that they are Gaussian functions. Even with simplifications, the process of implementing solutions using these assumptions can be complex and not always well-suited for practical applications.

The approach discussed in this section, called *Otsu's method* (Otsu [1979]), is an attractive alternative. The method is optimum in the sense that it maximizes the *between-class variance*, a well-known measure used in statistical discriminant analysis. The basic idea is that well-thresholded classes should be distinct with respect to the intensity values of their pixels and, conversely, that a threshold giving the best separation between classes in terms of their intensity values would be the best (optimum) threshold. In addition to its optimality, Otsu's method has the important property that it is based entirely on computations performed on the histogram of an image, an easily obtainable 1-D array.

Let $\{0, 1, 2, \ldots, L-1\}$ denote the L distinct intensity levels in a digital image of size $M \times N$ pixels, and let n_i denote the number of pixels with intensity i. The total number, MN, of pixels in the image is $MN = n_0 + n_1 + n_2 + \cdots + n_{L-1}$. The normalized histogram (see Section 3.3) has components $p_i = n_i/MN$, from which it follows that

$$\sum_{i=0}^{L-1} p_i = 1, \qquad p_i \geq 0 \tag{10.3-3}$$

Now, suppose that we select a threshold $T(k) = k$, $0 < k < L-1$, and use it to threshold the input image into two classes, C_1 and C_2, where C_1 consists of all the pixels in the image with intensity values in the range $[0, k]$ and C_2 consists of the pixels with values in the range $[k+1, L-1]$. Using this threshold, the probability, $P_1(k)$, that a pixel is assigned to (i.e., thresholded into) class C_1 is given by the cumulative sum

$$P_1(k) = \sum_{i=0}^{k} p_i \tag{10.3-4}$$

Viewed another way, this is the probability of class C_1 occurring. For example, if we set $k = 0$, the probability of class C_1 having any pixels assigned to it is zero. Similarly, the probability of class C_2 occurring is

$$P_2(k) = \sum_{i=k+1}^{L-1} p_i = 1 - P_1(k) \tag{10.3-5}$$

From Eq. (3.3-18), the mean intensity value of the pixels assigned to class C_1 is

$$\begin{aligned} m_1(k) &= \sum_{i=0}^{k} iP(i/C_1) \\ &= \sum_{i=0}^{k} iP(C_1/i)P(i)/P(C_1) \\ &= \frac{1}{P_1(k)} \sum_{i=0}^{k} ip_i \end{aligned} \tag{10.3-6}$$

where $P_1(k)$ is given in Eq. (10.3-4). The term $P(i/C_1)$ in the first line of Eq. (10.3-6) is the probability of value i, given that i comes from class C_1. The second line in the equation follows from Bayes' formula:

$$P(A/B) = P(B/A)P(A)/P(B)$$

The third line follows from the fact that $P(C_1/i)$, the probability of C_1 given i, is 1 because we are dealing only with values of i from class C_1. Also, $P(i)$ is the probability of the ith value, which is simply the ith component of the histogram, p_i. Finally, $P(C_1)$ is the probability of class C_1, which we know from Eq. (10.3-4) is equal to $P_1(k)$.

Similarly, the mean intensity value of the pixels assigned to class C_2 is

$$\begin{aligned} m_2(k) &= \sum_{i=k+1}^{L-1} iP(i/C_2) \\ &= \frac{1}{P_2(k)} \sum_{i=k+1}^{L-1} ip_i \end{aligned} \tag{10.3-7}$$

The cumulative mean (average intensity) up to level k is given by

$$m(k) = \sum_{i=0}^{k} ip_i \tag{10.3-8}$$

and the average intensity of the entire image (i.e., the *global* mean) is given by

$$m_G = \sum_{i=0}^{L-1} ip_i \tag{10.3-9}$$

The validity of the following two equations can be verified by direct substitution of the preceding results:

$$P_1 m_1 + P_2 m_2 = m_G \tag{10.3-10}$$

and

$$P_1 + P_2 = 1 \tag{10.3-11}$$

where we have omitted the ks temporarily in favor of notational clarity.

In order to evaluate the "goodness" of the threshold at level k we use the normalized, dimensionless metric

$$\eta = \frac{\sigma_B^2}{\sigma_G^2} \tag{10.3-12}$$

where σ_G^2 is the *global variance* [i.e., the intensity variance of all the pixels in the image, as given in Eq. (3.3-19)],

$$\sigma_G^2 = \sum_{i=0}^{L-1} (i - m_G)^2 p_i \tag{10.3-13}$$

and σ_B^2 is the *between-class variance*, defined as

$$\sigma_B^2 = P_1(m_1 - m_G)^2 + P_2(m_2 - m_G)^2 \tag{10.3-14}$$

This expression can be written also as

$$\begin{aligned} \sigma_B^2 &= P_1 P_2 (m_1 - m_2)^2 \\ &= \frac{(m_G P_1 - m)^2}{P_1(1 - P_1)} \end{aligned} \tag{10.3-15}$$

The second step in Eq. (10.3-15) makes sense only if P_1 is greater than 0 and less than 1, which, in view of Eq. (10.3-11), implies that P_2 must satisfy the same condition.

where m_G and m are as stated earlier. The first line of this equation follows from Eqs. (10.3-14), (10.3-10), and (10.3-11). The second line follows from Eqs. (10.3-5) through (10.3-9). This form is slightly more efficient computationally because the global mean, m_G, is computed only once, so only two parameters, m and P_1, need to be computed for any value of k.

We see from the first line in Eq. (10.3-15) that the farther the two means m_1 and m_2 are from each other the larger σ_B^2 will be, indicating that the between-class variance is a measure of *separability* between classes. Because σ_G^2 is a constant, it follows that η also is a measure of separability, and maximizing this metric is equivalent to maximizing σ_B^2. The objective, then, is to determine the threshold value, k, that maximizes the between-class variance, as stated at the beginning of this section. Note that Eq. (10.3-12) assumes implicitly that $\sigma_G^2 > 0$. This variance can be zero only when all the intensity levels in the image are the same, which implies the existence of only one class of pixels. This in turn means that $\eta = 0$ for a constant image since the separability of a single class from itself is zero.

Reintroducing k, we have the final results:

$$\eta(k) = \frac{\sigma_B^2(k)}{\sigma_G^2} \tag{10.3-16}$$

and

$$\sigma_B^2(k) = \frac{\left[m_G P_1(k) - m(k)\right]^2}{P_1(k)\left[1 - P_1(k)\right]} \tag{10.3-17}$$

Then, the optimum threshold is the value, k^*, that maximizes $\sigma_B^2(k)$:

$$\sigma_B^2(k^*) = \max_{0 \le k \le L-1} \sigma_B^2(k) \tag{10.3-18}$$

In other words, to find k^* we simply evaluate Eq. (10.3-18) for all *integer* values of k (such that the condition $0 < P_1(k) < 1$ holds) and select that value of k that yielded the maximum $\sigma_B^2(k)$. If the maximum exists for more than one value of k, it is customary to average the various values of k for which $\sigma_B^2(k)$ is maximum. It can be shown (Problem 10.33) that a maximum always exists, subject to the condition that $0 < P_1(k) < 1$. Evaluating Eqs. (10.3-17) and (10.3-18) for all values of k is a relatively inexpensive computational procedure, because the maximum number of integer values that k can have is L.

Once k^* has been obtained, the input image $f(x, y)$ is segmented as before:

$$g(x, y) = \begin{cases} 1 & \text{if } f(x, y) > k^* \\ 0 & \text{if } f(x, y) \le k^* \end{cases} \tag{10.3-19}$$

for $x = 0, 1, 2, \ldots, M - 1$ and $y = 0, 1, 2, \ldots, N - 1$. Note that all the quantities needed to evaluate Eq. (10.3-17) are obtained using only the histogram of $f(x, y)$. In addition to the optimum threshold, other information regarding the segmented image can be extracted from the histogram. For example, $P_1(k^*)$ and $P_2(k^*)$, the class probabilities evaluated at the optimum threshold, indicate the portions of the areas occupied by the classes (groups of pixels) in the thresholded image. Similarly, the means $m_1(k^*)$ and $m_2(k^*)$ are estimates of the average intensity of the classes in the original image.

The normalized metric η, evaluated at the optimum threshold value, $\eta(k^*)$, can be used to obtain a quantitative estimate of the separability of classes, which in turn gives an idea of the ease of thresholding a given image. This measure has values in the range

Although our interest is in the value of η at the optimum threshold, k^*, this inequality holds in general for any value of k in the range $[0, L - 1]$.

$$0 \le \eta(k^*) \le 1 \tag{10.3-20}$$

The lower bound is attainable only by images with a single, constant intensity level, as mentioned earlier. The upper bound is attainable only by 2-valued images with intensities equal to 0 and $L - 1$ (Problem 10.34).

Otsu's algorithm may be summarized as follows:

1. Compute the normalized histogram of the input image. Denote the components of the histogram by p_i, $i = 0, 1, 2, \ldots, L - 1$.
2. Compute the cumulative sums, $P_1(k)$, for $k = 0, 1, 2, \ldots, L - 1$, using Eq. (10.3-4).
3. Compute the cumulative means, $m(k)$, for $k = 0, 1, 2, \ldots, L - 1$, using Eq. (10.3-8).
4. Compute the global intensity mean, m_G, using (10.3-9).
5. Compute the between-class variance, $\sigma_B^2(k)$, for $k = 0, 1, 2, \ldots, L - 1$, using Eq. (10.3-17).
6. Obtain the Otsu threshold, k^*, as the value of k for which $\sigma_B^2(k)$ is maximum. If the maximum is not unique, obtain k^* by averaging the values of k corresponding to the various maxima detected.
7. Obtain the separability measure, η^*, by evaluating Eq. (10.3-16) at $k = k^*$.

The following example illustrates the preceding concepts.

EXAMPLE 10.16: Optimum global thresholding using Otsu's method.

■ Figure 10.39(a) shows an optical microscope image of polymersome cells, and Fig. 10.39(b) shows its histogram. The objective of this example is to segment the molecules from the background. Figure 10.39(c) is the result of using the basic global thresholding algorithm developed in the previous section. Because the histogram has no distinct valleys and the intensity difference between the background and objects is small, the algorithm failed to achieve the desired segmentation. Figure 10.39(d) shows the result obtained using Otsu's method. This result obviously is superior to Fig. 10.39(c). The threshold value computed by the basic algorithm was 169, while the threshold computed by Otsu's method was 181, which is closer to the lighter areas in the image defining the cells. The separability measure η was 0.467.

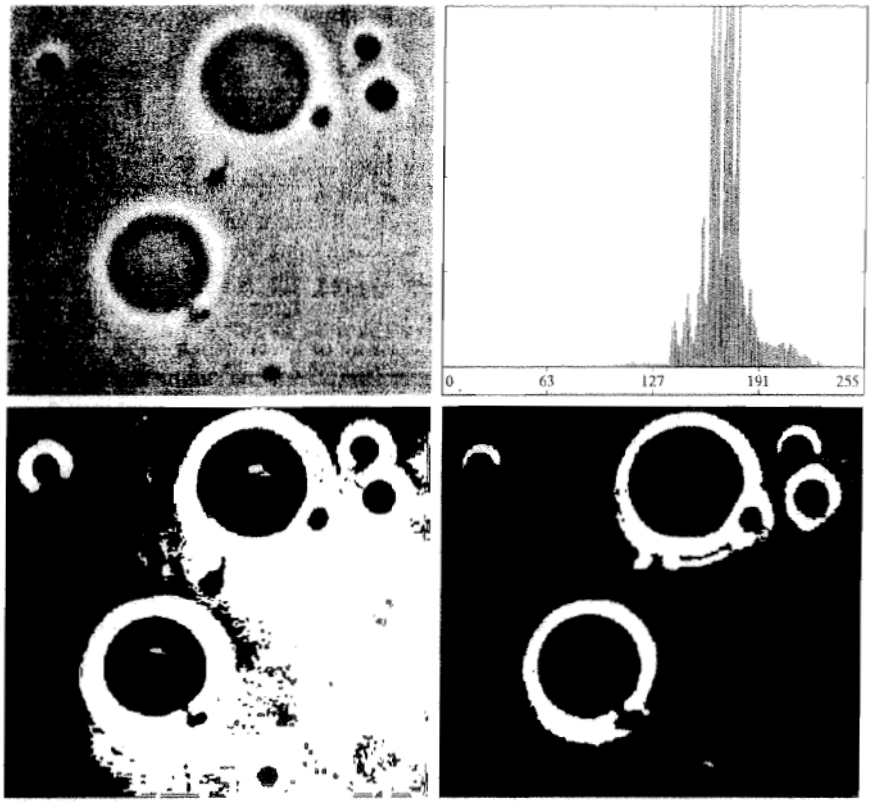
Polymersomes are cells artificially engineered using polymers. Polymersomes are invisible to the human immune system and can be used, for example, to deliver medication to targeted regions of the body.

As a point of interest, applying Otsu's method to the fingerprint image in Example 10.15 yielded a threshold of 125 and a separability measure of 0.944. The threshold is identical to the value (rounded to the nearest integer) obtained with the basic algorithm. This is not unexpected, given the nature of the histogram. In fact, the separability measure is high due primarily to the relatively large separation between modes and the deep valley between them. ■

10.3.4 Using Image Smoothing to Improve Global Thresholding

As noted in Fig. 10.36, noise can turn a simple thresholding problem into an unsolvable one. When noise cannot be reduced at the source, and thresholding is the segmentation method of choice, a technique that often enhances performance is to smooth the image prior to thresholding. We illustrate the approach with an example.

Figure 10.40(a) is the image from Fig. 10.36(c), Fig. 10.40(b) shows its histogram, and Fig. 10.40(c) is the image thresholded using Otsu's method. Every black point in the white region and every white point in the black region is a

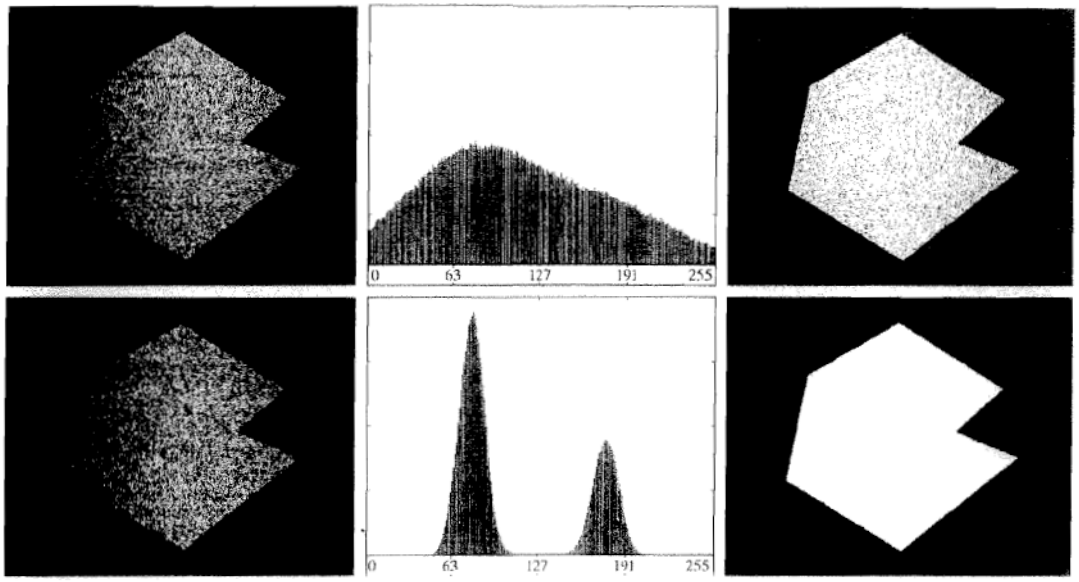


a b
c d

FIGURE 10.39
(a) Original image.
(b) Histogram (high peaks were clipped to highlight details in the lower values).
(c) Segmentation result using the basic global algorithm from Section 10.3.2.
(d) Result obtained using Otsu's method.
(Original image courtesy of Professor Daniel A. Hammer, the University of Pennsylvania.)

thresholding error, so the segmentation was highly unsuccessful. Figure 10.40(d) shows the result of smoothing the noisy image with an averaging mask of size 5×5 (the image is of size 651×814 pixels), and Fig. 10.40(e) is its histogram. The improvement in the shape of the histogram due to smoothing is evident, and we would expect thresholding of the smoothed image to be nearly perfect. As Fig. 10.40(f) shows, this indeed was the case. The slight distortion of the boundary between object and background in the segmented, smoothed image was caused by the blurring of the boundary. In fact, the more aggressively we smooth an image, the more boundary errors we should anticipate in the segmented result.

Next we consider the effect of reducing the size of the region in Fig. 10.40(a) with respect to the background. Figure 10.41(a) shows the result. The noise in this image is additive Gaussian noise with zero mean and a standard deviation of 10 intensity levels (as opposed to 50 in the previous example). As Fig. 10.41(b) shows, the histogram has no clear valley, so we would expect segmentation to fail, a fact that is confirmed by the result in Fig. 10.41(c). Figure 10.41(d) shows the image smoothed with an averaging mask of size 5×5, and Fig. 10.40(e) is the corresponding histogram. As expected, the net effect was to reduce the spread of the histogram, but the distribution still is unimodal. As Fig. 10.40(f) shows, segmentation failed again. The reason for the failure can be traced to the fact that the region is so small that its contribution to the histogram is insignificant compared to the intensity spread caused by noise. In



a b c
d e f

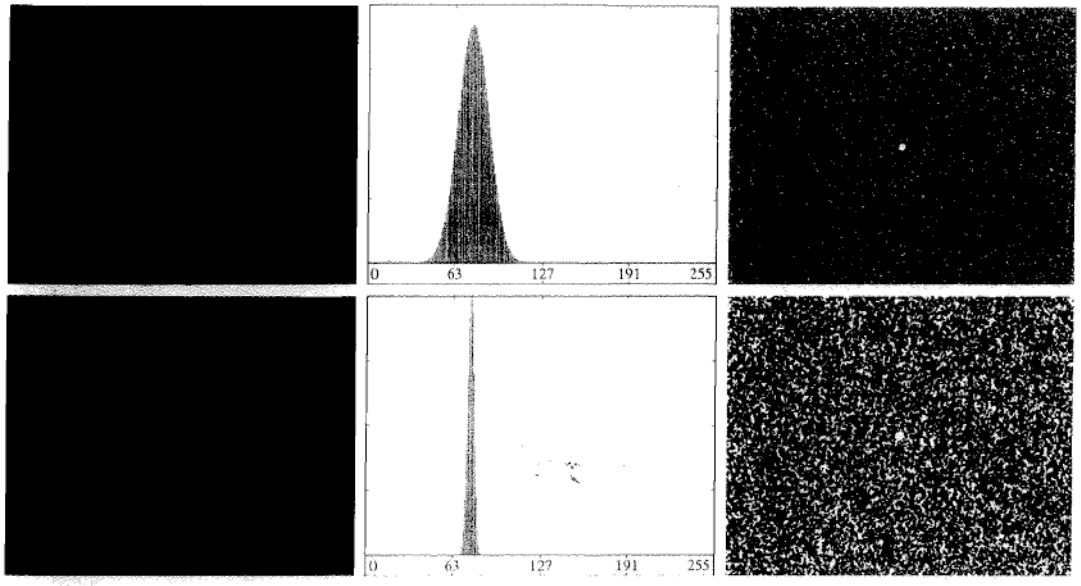
FIGURE 10.40 (a) Noisy image from Fig. 10.36 and (b) its histogram. (c) Result obtained using Otsu's method. (d) Noisy image smoothed using a 5×5 averaging mask and (e) its histogram. (f) Result of thresholding using Otsu's method.

situations such as this, the approach discussed in the following section is more likely to succeed.

10.3.5 Using Edges to Improve Global Thresholding

Based on the discussion in the previous four sections, we conclude that the chances of selecting a "good" threshold are enhanced considerably if the histogram peaks are tall, narrow, symmetric, and separated by deep valleys. One approach for improving the shape of histograms is to consider only those pixels that lie on or near the edges between objects and the background. An immediate and obvious improvement is that histograms would be less dependent on the relative sizes of objects and the background. For instance, the histogram of an image composed of a small object on a large background area (or vice versa) would be dominated by a large peak because of the high concentration of one type of pixels. We saw in the previous section that this can lead to failure in thresholding.

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



a b c
d e f

FIGURE 10.41 (a) Noisy image and (b) its histogram. (c) Result obtained using Otsu's method. (d) Noisy image smoothed using a 5×5 averaging mask and (e) its histogram. (f) Result of thresholding using Otsu's method. Thresholding failed in both cases.

The approach just discussed assumes that the edges between objects and background are known. This information clearly is not available during segmentation, as finding a division between objects and background is precisely what segmentation is all about. However, with reference to the discussion in Section 10.2, an indication of whether a pixel is on an edge may be obtained by computing its gradient or Laplacian. For example, the average value of the Laplacian is 0 at the transition of an edge (see Fig. 10.10), so the valleys of histograms formed from the pixels selected by a Laplacian criterion can be expected to be sparsely populated. This property tends to produce the desirable deep valleys discussed above. In practice, comparable results typically are obtained using either the gradient or Laplacian images, with the latter being favored because it is computationally more attractive and is also an isotropic edge detector.

The preceding discussion is summarized in the following algorithm, where $f(x, y)$ is the input image:

1. Compute an edge image as either the magnitude of the gradient, or absolute value of the Laplacian, of $f(x, y)$ using any of the methods discussed in Section 10.2.
2. Specify a threshold value, T.
3. Threshold the image from Step 1 using the threshold from Step 2 to produce a binary image, $g_T(x, y)$. This image is used as a *mask image* in the following step to select pixels from $f(x, y)$ corresponding to "strong" edge pixels.

It is possible to modify this algorithm so that both the magnitude of the gradient and the absolute value of the Laplacian images are used. In this case, we would specify a threshold for each image and form the logical OR of the two results to obtain the marker image. This approach is useful when more control is desired over the points deemed to be valid edge points.

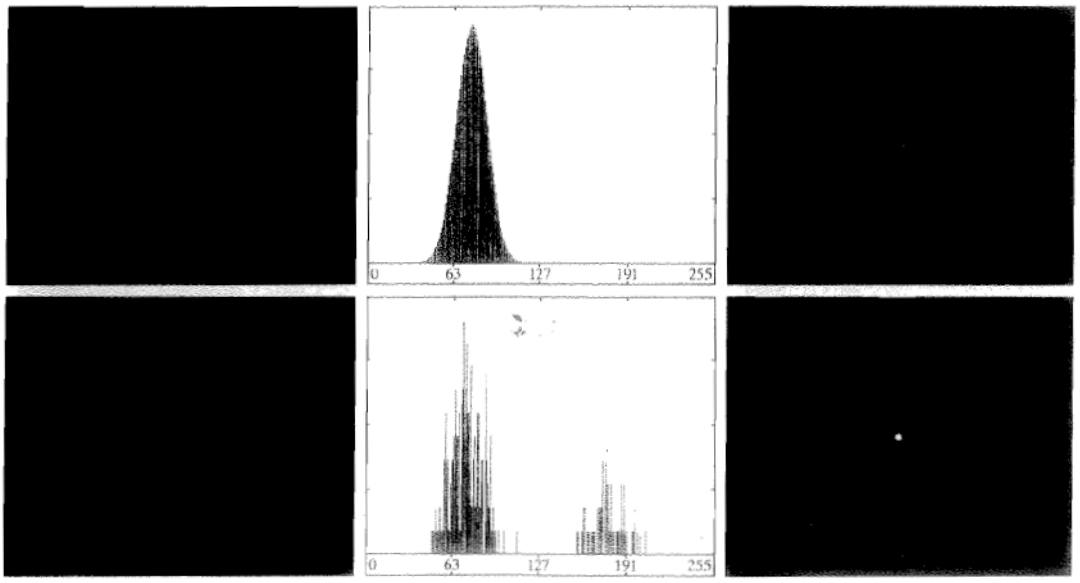
4. Compute a histogram using only the pixels in $f(x, y)$ that correspond to the locations of the 1-valued pixels in $g_T(x, y)$.
5. Use the histogram from Step 4 to segment $f(x, y)$ globally using, for example, Otsu's method.

If T is set to any value less than the minimum value of the edge image then, according to Eq. (10.3-1), $g_T(x, y)$ will consist of all 1s, implying that all pixels of $f(x, y)$ will be used to compute the image histogram. In this case, the preceding algorithm becomes global thresholding in which the histogram of the original image is used. It is customary to specify the value of T corresponding to a percentile, which typically is set high (e.g., in the high 90s) so that few pixels in the gradient/Laplacian image will be used in the computation. The following examples illustrate the concepts just discussed. The first example uses the gradient and the second uses the Laplacian. Similar results can be obtained in both examples using either approach. The important issue is to generate a suitable derivative image.

The nth percentile is the smallest number that is greater than n% of the numbers in a given set. For example, if you received a 95 in a test and this score was greater than 85% of all the students taking the test, then you would be in the 85th percentile with respect to the test scores.

EXAMPLE 10.17: Using edge information based on the gradient to improve global thresholding.

■ Figures 10.42(a) and (b) show the image and histogram from Fig. 10.41. You saw that this image could not be segmented by smoothing followed by thresholding. The objective of this example is to solve the problem using edge information. Figure 10.42(c) is the gradient magnitude image thresholded at the



a b c
d e f

FIGURE 10.42 (a) Noisy image from Fig. 10.41(a) and (b) its histogram. (c) Gradient magnitude image thresholded at the 99.7 percentile. (d) Image formed as the product of (a) and (c). (e) Histogram of the nonzero pixels in the image in (d). (f) Result of segmenting image (a) with the Otsu threshold based on the histogram in (c). The threshold was 134, which is approximately midway between the peaks in this histogram.

99.7 percentile. Figure 10.42(d) is the image formed by multiplying this (mask) image by the input image. Figure 10.42(e) is the histogram of the nonzero elements in Fig. 10.42(d). Note that this histogram has the important features discussed earlier; that is, it has reasonably symmetrical modes separated by a deep valley. Thus, while the histogram of the original noisy image offered no hope for successful thresholding, the histogram in Fig. 10.42(e) indicates that thresholding of the small object from the background is indeed possible. The result in Fig. 10.42(f) shows that indeed this is the case. This image was obtained by using Otsu's method to obtain a threshold based on the histogram in Fig. 10.42(e) and then applying this threshold globally to the noisy image in Fig. 10.42(a). The result is nearly perfect. ■

EXAMPLE 10.18: Using edge information based on the Laplacian to improve global thresholding.

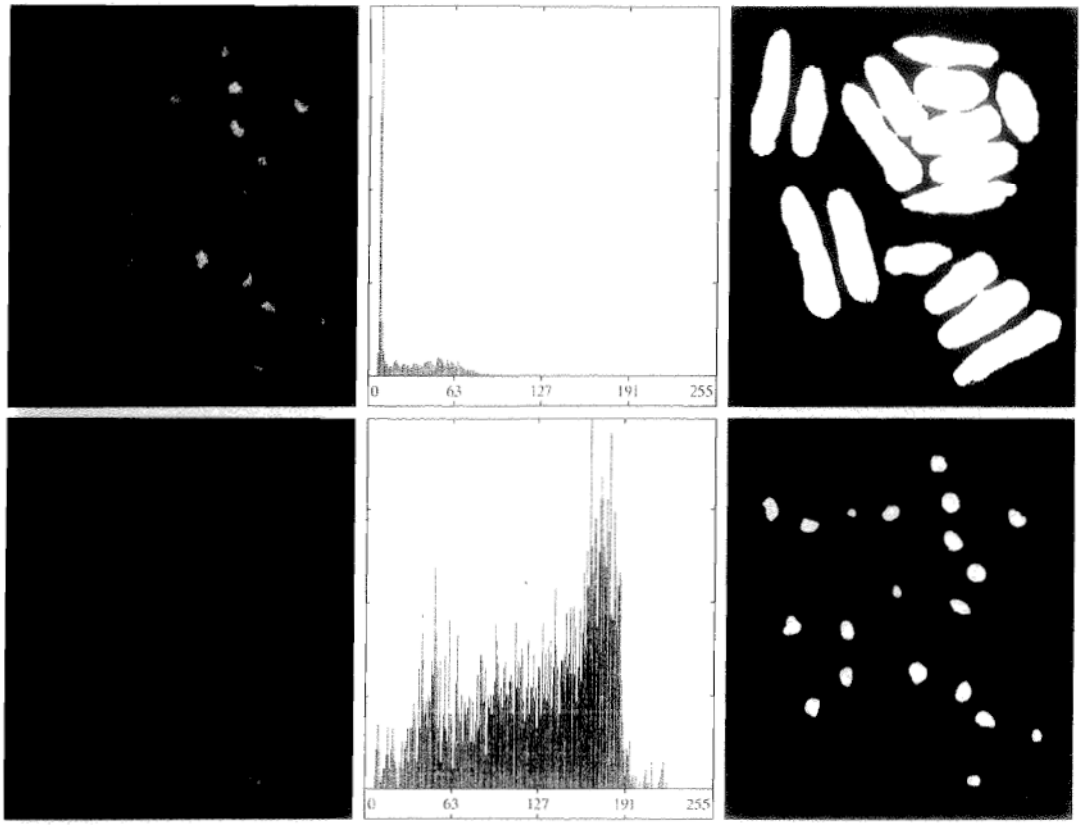
■ In this example we consider a more complex thresholding problem. Figure 10.43(a) shows an 8-bit image of yeast cells in which we wish to use global thresholding to obtain the regions corresponding to the bright spots. As a starting point, Fig. 10.43(b) shows the image histogram, and Fig. 10.43(c) is the result obtained using Otsu's method directly on the image, using the histogram shown. We see that Otsu's method failed to achieve the original objective of detecting the bright spots, and, although the method was able to isolate some of the cell regions themselves, several of the segmented regions on the right are not disjoint. The threshold computed by the Otsu method was 42 and the separability measure was 0.636.

Figure 10.43(d) shows the image $g_T(x, y)$ obtained by computing the absolute value of the Laplacian image and then thresholding it with T set to 115 on an intensity scale in the range $[0, 255]$. This value of T corresponds approximately to the 99.5 percentile of the values in the absolute Laplacian image, so thresholding at this level should result in a sparse set of pixels, as Fig. 10.43(d) shows. Note in this image how the points cluster near the edges of the bright spots, as expected from the preceding discussion. Figure 10.43(e) is the histogram of the nonzero pixels in the product of (a) and (d). Finally, Fig. 10.43(f) shows the result of globally segmenting the original image using Otsu's method based on the histogram in Fig. 10.43(e). This result agrees with the locations of the bright spots in the image. The threshold computed by the Otsu method was 115 and the separability measure was 0.762, both of which are higher than the values obtained by using the original histogram.

By varying the percentile at which the threshold is set we can even improve on the segmentation of the cell regions. For example, Fig. 10.44 shows the result obtained using the same procedure as in the previous paragraph, but with the threshold set at 55, which is approximately 5% of the maximum value of the absolute Laplacian image. This value is at the 53.9 percentile of the values in that image. This result clearly is superior to the result in Fig. 10.43(c) obtained using Otsu's method with the histogram of the original image. ■

10.3.6 Multiple Thresholds

Thus far, we have focused attention on image segmentation using a single global threshold. The thresholding method introduced in Section 10.3.3 can be extended to an arbitrary number of thresholds, because the separability measure



a b c
d e f

FIGURE 10.43 (a) Image of yeast cells. (b) Histogram of (a). (c) Segmentation of (a) with Otsu's method using the histogram in (b). (d) Thresholded absolute Laplacian. (e) Histogram of the nonzero pixels in the product of (a) and (d). (f) Original image thresholded using Otsu's method based on the histogram in (e). (Original image courtesy of Professor Susan L. Forsburg, University of Southern California.)

FIGURE 10.44 Image in Fig. 10.43(a) segmented using the same procedure as explained in Figs. 10.43(d)–(f), but using a lower value to threshold the absolute Laplacian image.

on which it is based also extends to an arbitrary number of classes (Fukunaga [1972]). In the case of K classes, $C_1, C_2, \ldots C_K$, the between-class variance generalizes to the expression

$$\sigma_B^2 = \sum_{k=1}^{K} P_k (m_k - m_G)^2 \tag{10.3-21}$$

where

$$P_k = \sum_{i \in C_k} p_i \tag{10.3-22}$$

$$m_k = \frac{1}{P_k} \sum_{i \in C_k} i p_i \tag{10.3-23}$$

and m_G is the global mean given in Eq. (10.3-9). The K classes are separated by $K - 1$ thresholds whose values, $k_1^*, k_2^*, \ldots, k_{K-1}^*$, are the values that maximize Eq. (10.3-21):

$$\sigma_B^2(k_1^*, k_2^*, \ldots, k_{K-1}^*) = \max_{0 < k_1 < k_2 < \ldots k_{n-1} < L-1} \sigma_B^2(k_1, k_2, \ldots k_{K-1}) \tag{10.3-24}$$

Although this result is perfectly general, it begins to lose meaning as the number of classes increases, because we are dealing with only one variable (intensity). In fact, the between-class variance usually is cast in terms of multiple variables expressed as vectors (Fukunaga [1972]). In practice, using multiple global thresholding is considered a viable approach when there is reason to believe that the problem can be solved effectively with two thresholds. Applications that require more than two thresholds generally are solved using more than just intensity values. Instead, the approach is to use additional descriptors (e.g., color) and the application is cast as a pattern recognition problem, as explained in Section 10.3.8.

Thresholding with two thresholds sometimes is referred to as *hysteresis thresholding*.

For three classes consisting of three intensity intervals (which are separated by two thresholds) the between-class variance is given by:

$$\sigma_B^2 = P_1(m_1 - m_G)^2 + P_2(m_2 - m_G)^2 + P_3(m_3 - m_G)^2 \tag{10.3-25}$$

where

$$\begin{aligned} P_1 &= \sum_{i=0}^{k_1} p_i \\ P_2 &= \sum_{i=k_1+1}^{k_2} p_i \\ P_3 &= \sum_{i=k_2+1}^{L-1} p_i \end{aligned} \tag{10.3-26}$$

and

$$m_1 = \frac{1}{P_1}\sum_{i=0}^{k_1} ip_i$$

$$m_2 = \frac{1}{P_2}\sum_{i=k_1+1}^{k_2} ip_i \tag{10.3-27}$$

$$m_3 = \frac{1}{P_3}\sum_{i=k_2+1}^{L-1} ip_i$$

As in Eqs. (10.3-10) and (10.3-11), the following relationships hold:

$$P_1m_1 + P_2m_2 + P_3m_3 = m_G \tag{10.3-28}$$

and

$$P_1 + P_2 + P_3 = 1 \tag{10.3-29}$$

We see that the P and m terms and, therefore σ_B^2, are functions of k_1 and k_2. The two optimum threshold values, k_1^* and k_2^*, are the values that maximize $\sigma_B^2(k_1, k_2)$. In other words, as in the single-threshold case discussed in Section 10.3.3, we find the optimum thresholds by finding

$$\sigma_B^2(k_1^*, k_2^*) = \max_{0<k_1<k_2<L-1} \sigma_B^2(k_1, k_2) \tag{10.3-30}$$

The procedure starts by selecting the first value of k_1 (that value is 1 because looking for a threshold at 0 intensity makes no sense; also, keep in mind that the increment values are integers because we are dealing with intensities). Next, k_2 is incremented through all its values greater than k_1 and less than $L - 1$ (i.e., $k_2 = k_1 + 1, \ldots, L - 2$). Then k_1 is incremented to its next value and k_2 is incremented again through all its values greater than k_1. This procedure is repeated until $k_1 = L - 3$. The result of this process is a 2-D array, $\sigma_B^2(k_1, k_2)$, and the last step is to look for the maximum value in this array. The values of k_1 and k_2 corresponding to that maximum are the optimum thresholds, k_1^* and k_2^*. If there are several maxima, the corresponding values of k_1 and k_2 are averaged to obtain the final thresholds. The thresholded image is then given by

$$g(x, y) = \begin{cases} a & \text{if } f(x, y) \le k_1^* \\ b & \text{if } k_1^* < f(x, y) \le k_2^* \\ c & \text{if } f(x, y) > k_2^* \end{cases} \tag{10.3-31}$$

where a, b, and c are any three valid intensity values.

Finally, we note that the separability measure defined in Section 10.3.3 for one threshold extends directly to multiple thresholds:

$$\eta(k_1^*, k_2^*) = \frac{\sigma_B^2(k_1^*, k_2^*)}{\sigma_G^2} \tag{10.3-32}$$

where σ_G^2 is the total image variance from Eq. (10.3-13).

EXAMPLE 10.19: Multiple global thresholding.

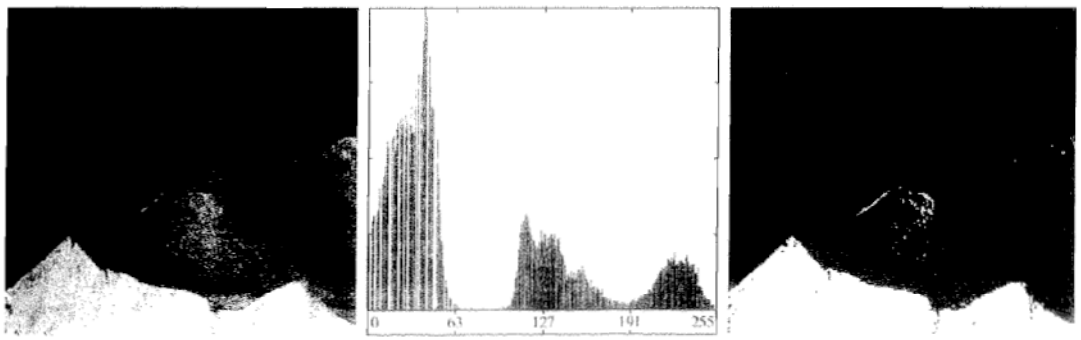
■ Figure 10.45(a) shows an image of an iceberg. The objective of this example is to segment the image into three regions: the dark background, the illuminated area of the iceberg, and the area in shadows. It is evident from the image histogram in Fig. 10.45(b) that two thresholds are required to solve this problem. The procedure discussed above resulted in the thresholds $k_1^* = 80$ and $k_2^* = 177$, which we note from Fig. 10.45(b) are near the centers of the two histogram valleys. Figure 10.45(c) is the segmentation that resulted using these two thresholds in Eq. (10.3-31). The separability measure was 0.954. The principal reason this example worked out so well can be traced to the histogram having three distinct modes separated by reasonably wide, deep valleys. ■

10.3.3 Variable Thresholding

As discussed in Section 10.3.1, factors such as noise and nonuniform illumination play a major role in the performance of a thresholding algorithm. We showed in Sections 10.3.4 and 10.3.5 that image smoothing and using edge information can help significantly. However, it frequently is the case that this type of preprocessing is either impractical or simply ineffective in improving the situation to the point where the problem is solvable by any of the methods discussed thus far. In such situations, the next level of thresholding complexity involves variable thresholding. In this section, we discuss various techniques for choosing variable thresholds.

Image partitioning

One of the simplest approaches to variable thresholding is to subdivide an image into nonoverlapping rectangles. This approach is used to compensate for non-uniformities in illumination and/or reflectance. The rectangles are chosen small enough so that the illumination of each is approximately uniform. We illustrate this approach with an example.



a b c

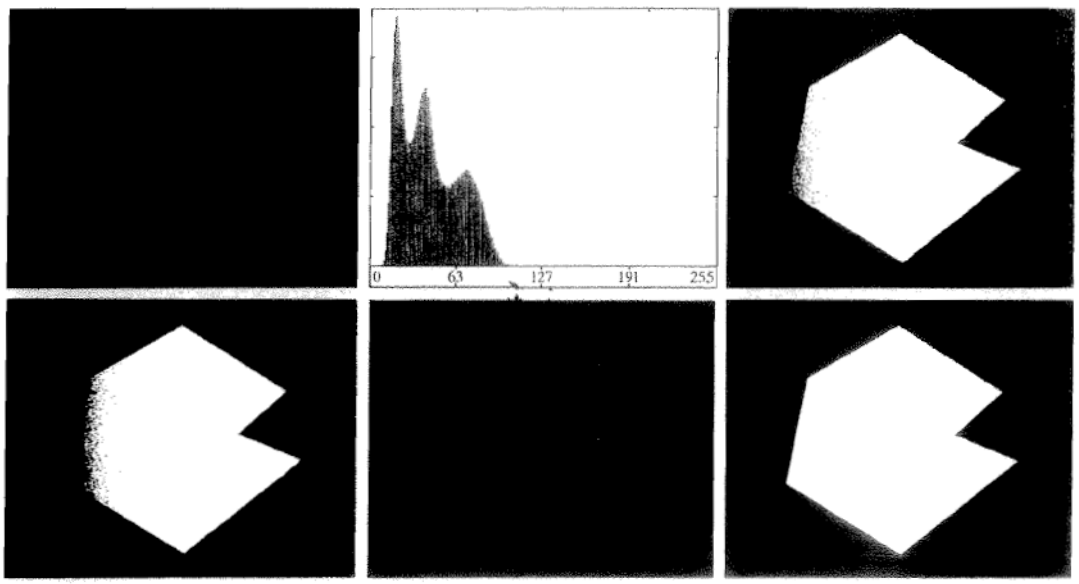
FIGURE 10.45 (a) Image of iceberg. (b) Histogram. (c) Image segmented into three regions using dual Otsu thresholds. (Original image courtesy of NOAA.)

EXAMPLE 10.20: Variable thresholding via image partitioning.

■ Figure 10.46(a) shows the image from Fig. 10.37(c), and Fig. 10.46(b) shows its histogram. When discussing Fig. 10.37(c) we concluded that this image could not be segmented with a global threshold, a fact confirmed by Figs. 10.46(c) and (d), which show the results of segmenting the image using the iterative scheme discussed in Section 10.3.2 and Otsu's method, respectively. Both methods produced comparable results, in which numerous segmentation errors are visible.

Figure 10.46(e) shows the original image subdivided into six rectangular regions, and Fig. 10.46(f) is the result of applying Otsu's global method to each subimage. Although some errors in segmentation are visible, image subdivision produced a reasonable result on an image that is quite difficult to segment. The reason for the improvement is explained easily by analyzing the histogram of each subimage. As Fig. 10.47 shows, each subimage is characterized by a bimodal histogram with a deep valley between the modes, a fact that we know will lead to effective global thresholding.

Image subdivision generally works well when the objects of interest and the background occupy regions of reasonably comparable size, as in Fig. 10.46. When this is not the case, the method typically fails because of the likelihood of subdivisions containing only object or background pixels. Although this situation can be addressed by using additional techniques to determine when a subdivision contains both types of pixels, the logic required to address different



a b c
d e f

FIGURE 10.46 (a) Noisy, shaded image and (b) its histogram. (c) Segmentation of (a) using the iterative global algorithm from Section 10.3.2. (d) Result obtained using Otsu's method. (e) Image subdivided into six subimages. (f) Result of applying Otsu's method to each subimage individually.

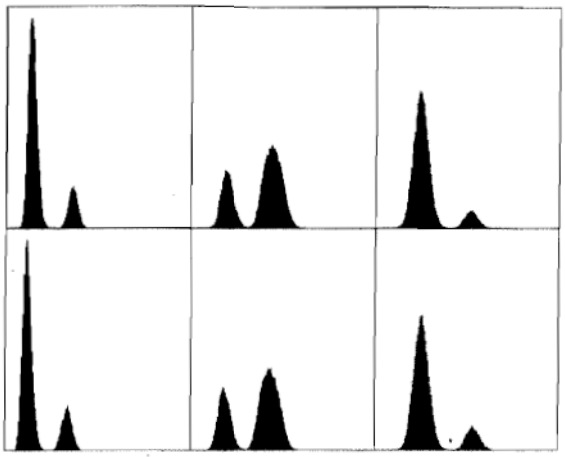


FIGURE 10.47 Histograms of the six subimages in Fig. 10.46(e).

scenarios can get complicated. In such situations, methods such as those discussed in the remainder of this section typically are preferable. ■

Variable thresholding based on local image properties

A more general approach than the image subdivision method discussed in the previous section is to compute a threshold at every point, (x, y), in the image based on one or more specified properties computed in a neighborhood of (x, y). Although this may seem like a laborious process, modern algorithms and hardware allow for fast neighborhood processing, especially for common functions such as logical and arithmetic operations.

We illustrate the basic approach to local thresholding using the standard deviation and mean of the pixels in a neighborhood of every point in an image. These two quantities are quite useful for determining local thresholds because they are descriptors of local contrast and average intensity. Let σ_{xy} and m_{xy} denote the standard deviation and mean value of the set of pixels contained in a neighborhood, S_{xy}, centered at coordinates (x, y) in an image (see Section 3.3.4 regarding computation of the local mean and standard deviation). The following are common forms of variable, local thresholds:

$$T_{xy} = a\sigma_{xy} + bm_{xy} \tag{10.3-33}$$

where a and b are nonnegative constants, and

$$T_{xy} = a\sigma_{xy} + bm_G \tag{10.3-34}$$

where m_G is the global image mean. The segmented image is computed as

$$g(x, y) = \begin{cases} 1 & \text{if } f(x, y) > T_{xy} \\ 0 & \text{if } f(x, y) \leq T_{xy} \end{cases} \tag{10.3-35}$$

where $f(x, y)$ is the input image. This equation is evaluated for all pixel locations in the image, and a different threshold is computed at each location (x, y) using the pixels in the neighborhood S_{xy}.

Significant power (with a modest increase in computation) can be added to local thresholding by using predicates based on the parameters computed in the neighborhoods of (x, y):

$$g(x, y) = \begin{cases} 1 & \text{if } Q(\text{local parameters}) \text{ is true} \\ 0 & \text{if } Q(\text{local parameters}) \text{ is false} \end{cases} \tag{10.3-36}$$

where Q is a *predicate* based on parameters computed using the pixels in neighborhood S_{xy}. For example, consider the following predicate, $Q(\sigma_{xy}, m_{xy})$, based on the local mean and standard deviation:

$$Q(\sigma_{xy}, m_{xy}) = \begin{cases} \text{true} & \text{if } f(x, y) > a\sigma_{xy} \text{ AND } f(x, y) > bm_{xy} \\ \text{false} & \text{otherwise} \end{cases} \tag{10.3-37}$$

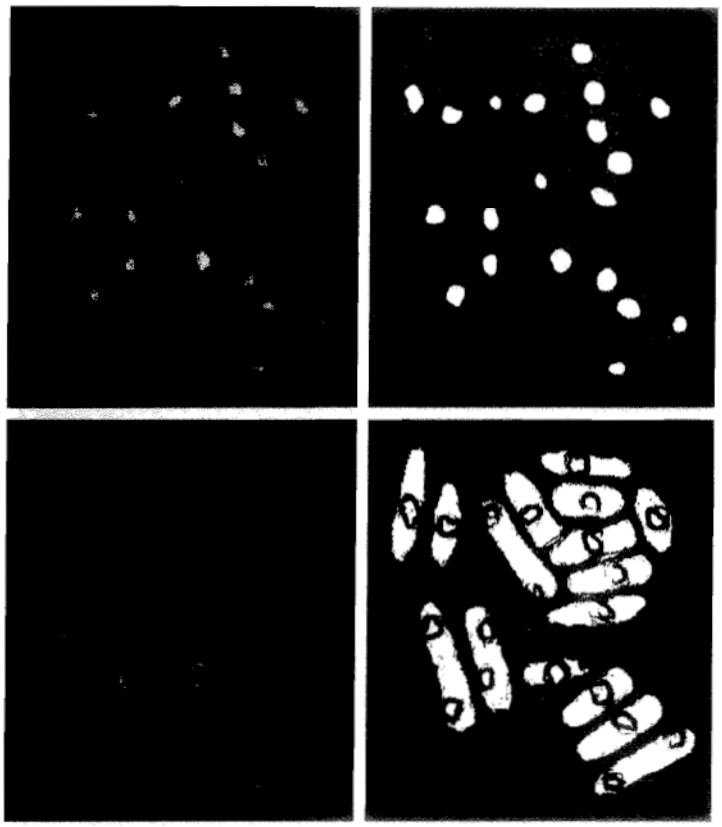
Note that Eq. (10.3-35) is a special case of Eq. (10.3-36), obtained by letting Q be true if $f(x, y) > T_{xy}$ and false otherwise. In this case, the predicate is based simply on the intensity at a point.

EXAMPLE 10.21: Variable thresholding based on local image properties.

■ Figure 10.48(a) shows the yeast image from Example 10.18. This image has three predominant intensity levels, so it is reasonable to assume that perhaps dual thresholding could be a good segmentation approach. Figure 10.48(b) is the result of using the dual thresholding method explained in Section 10.3.6. As the figure shows, it was possible to isolate the bright areas from the background, but the mid-gray regions on the right side of the image were not segmented properly (recall that we encountered a similar problem with Fig. 10.43(c) in Example 10.18). To illustrate the use of local thresholding, we computed the local standard deviation σ_{xy} for all (x, y) in the input image using a neighborhood of size 3×3. Figure 10.48(c) shows the result. Note how the faint outer lines correctly delineate the boundaries of the cells. Next, we formed a predicate of the form shown in Eq. (10.3-37) but using the global mean instead of m_{xy}. Choosing the global mean generally gives better results when the background is nearly constant and all the object intensities are above or below the background intensity. The values $a = 30$ and $b = 1.5$ were used in completing the specification of the predicate (these values were determined experimentally, as is usually the case in applications such as this). The image was then segmented using Eq. (10.3-36). As Fig. 10.48(d) shows, the result agrees quite closely with the two types of intensity regions prevalent in the input image. Note in particular that all the outer regions were segmented properly and that most of the inner, brighter regions were isolated correctly. ■

Using moving averages

A special case of the local thresholding method just discussed is based on computing a moving average along scan lines of an image. This implementation is quite useful in document processing, where speed is a fundamental requirement. The scanning typically is carried out line by line in a zigzag pattern to



a b
c d

FIGURE 10.48
(a) Image from Fig. 10.43. (b) Image segmented using the dual thresholding approach discussed in Section 10.3.6. (c) Image of local standard deviations. (d) Result obtained using local thresholding.

reduce illumination bias. Let z_{k+1} denote the intensity of the point encountered in the scanning sequence at step $k + 1$. The moving average (mean intensity) at this new point is given by

$$\begin{aligned} m(k+1) &= \frac{1}{n}\sum_{i=k+2-n}^{k+1} z_i \\ &= m(k) + \frac{1}{n}(z_{k+1} - z_{k-n}) \end{aligned} \tag{10.3-38}$$

The first expression is valid for $k \geq n - 1$. When k is less than $n - 1$, averages are formed with the available points. Similarly, the second expression is valid for $k \geq n + 1$.

where n denotes the number of points used in computing the average and $m(1) = z_1/n$. This initial value is not strictly correct because the average of a single point is the value of the point itself. However, we use $m(1) = z_1/n$ so that no special computations are required when Eq. (10.3-38) first starts up. Another way of viewing it is that this is the value we would obtain if the border of the image were padded with $n - 1$ zeros. The algorithm is initialized only once, not at every row. Because a moving average is computed for every point in the image, segmentation is implemented using Eq. (10.3-35) with $T_{xy} = bm_{xy}$ where b is constant and m_{xy} is the moving average from Eq. (10.3-38) at point (x, y) in the input image.

EXAMPLE 10.22: Document thresholding using moving averages.

■ Figure 10.49(a) shows an image of handwritten text shaded by a spot intensity pattern. This form of intensity shading is typical of images obtained with a photographic flash. Figure 10.49(b) is the result of segmentation using the Otsu global thresholding method. It is not unexpected that global thresholding could not overcome the intensity variation. Figure 10.49(c) shows successful segmentation with local thresholding using moving averages. A rule of thumb is to let n equal 5 times the average stroke width. In this case, the average width was 4 pixels, so we let $n = 20$ in Eq. (10.3-38) and used $b = 0.5$.

As another illustration of the effectiveness of this segmentation approach we used the same parameters as in the previous paragraph to segment the image in Fig. 10.50(a), which is corrupted by a sinusoidal intensity variation typical of the variation that may occur when the power supply in a document scanner is not grounded properly. As Figs. 10.50(b) and (c) show, the segmentation results are comparable to those in Fig. 10.49.

It is of interest to note that successful segmentation results were obtained in both cases using the same values for n and b, which shows the relative ruggedness of the approach. In general, thresholding based on moving averages works well when the objects of interest are small (or thin) with respect to the image size, a condition satisfied by images of typed or handwritten text. ■

10.3.8 Multivariable Thresholding

Thus far, we have been concerned with thresholding based on a single variable: gray-scale intensity. In some cases, a sensor can make available more than one variable to characterize each pixel in an image, and thus allow *multivariable thresholding*. A notable example is color imaging, where red (R), green (G), and blue (B) components are used to form a composite color image (see Chapter 6). In this case, each "pixel" is characterized by three values, and can be represented as a 3-D vector, $\mathbf{z} = (z_1, z_2, z_3)^T$, whose components are the RGB colors at a point. These 3-D points often are referred to as *voxels*, to denote *volumetric* elements, as opposed to *image* elements.

a b c

FIGURE 10.49 (a) Text image corrupted by spot shading. (b) Result of global thresholding using Otsu's method. (c) Result of local thresholding using moving averages.

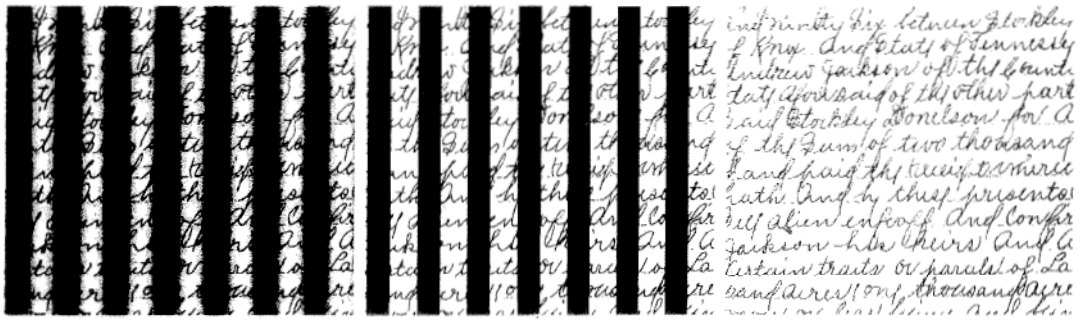
As discussed in some detail in Section 6.7, multivariable thresholding may be viewed as a distance computation. Suppose that we want to extract from a color image all regions having a specified color range: say, reddish hues. Let **a** denote the average reddish color in which we are interested. One way to segment a color image based on this parameter is to compute a distance measure, $D(\mathbf{z}, \mathbf{a})$, between an arbitrary color point, $\mathbf{z}$, and the average color, $\mathbf{a}$. Then, we segment the input image as follows:

$$g = \begin{cases} 1 & \text{if } D(\mathbf{z}, \mathbf{a}) < T \\ 0 & \text{otherwise} \end{cases} \tag{10.3-39}$$

where T is a threshold, and it is understood that the distance computation is performed at all coordinates in the input image to generate the corresponding segmented values in g. Note that the inequalities in this equation are the opposite of the inequalities we used in Eq. (10.3-1) for thresholding a single variable. The reason is that the equation $D(\mathbf{z}, \mathbf{a}) = T$ defines a volume (see Fig. 6.43) and it is more intuitive to think of segmented pixel values as being contained within the volume and background pixel values as being on the surface or outside the volume. Equation (10.3-39) reduces to Eq. (10.3-1) by letting $D(\mathbf{z}, \mathbf{a}) = -f(x, y)$.

Observe that the condition $f(x, y) > T$ basically says that the Euclidean distance between the value of f and the origin of the real line exceeds the value of T. Thus, thresholding is based on the computation of a distance measure, and the form of Eq. (10.3-39) depends on the measure used. If, in general, $\mathbf{z}$ in an n-dimensional vector, we know from Section 2.6.6 that the n-dimensional *Euclidean distance* is defined as

$$\begin{aligned} D(\mathbf{z}, \mathbf{a}) &= \|\mathbf{z} - \mathbf{a}\| \\ &= \left[(\mathbf{z} - \mathbf{a})^T(\mathbf{z} - \mathbf{a})\right]^{\frac{1}{2}} \end{aligned} \tag{10.3-40}$$



a b c

FIGURE 10.50 (a) Text image corrupted by sinusoidal shading. (b) Result of global thresholding using Otsu's method. (c) Result of local thresholding using moving averages.

The equation $D(\mathbf{z}, \mathbf{a}) = T$ describes a sphere (called a *hypersphere*) in n-dimensional Euclidean space (Fig. 6.43 shows a 3-D example). A more powerful distance measure is the so-called *Mahalanobis distance*, defined as

$$D(\mathbf{z}, \mathbf{a}) = \left[(\mathbf{z} - \mathbf{a})^T \mathbf{C}^{-1} (\mathbf{z} - \mathbf{a})\right]^{\frac{1}{2}} \tag{10.3-41}$$

where $\mathbf{C}$ is the covariance matrix of the $\mathbf{z}$s, as discussed Section 12.2.2. $D(\mathbf{z}, \mathbf{a}) = T$ describes an n-dimensional hyperellipse (Fig. 6.43 shows a 3-D example). This expression reduces to Eq. (10.3-40) when $\mathbf{C} = \mathbf{I}$, the identity matrix.

We gave a detailed example in Section 6.7 regarding the use of these expressions. We also discuss in Section 12.2 the problem of segmenting regions out of an image using pattern recognition techniques based on decision functions, which may be viewed as a multiclass, multivariable thresholding problem.

10.4 Region-Based Segmentation

You should review the terminology introduced in Section 10.1 before proceeding.

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

10.4.1 Region Growing

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

Selecting a set of one or more starting points often can be based on the nature of the problem, as shown later in Example 10.23. When a priori information is not available, the procedure is to compute at every pixel the same set of properties that ultimately will be used to assign pixels to regions during the growing process. If the result of these computations shows clusters of values, the pixels whose properties place them near the centroid of these clusters can be used as seeds.

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

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

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

Let: $f(x, y)$ denote an input image array; $S(x, y)$ denote a *seed* array containing 1s at the locations of seed points and 0s elsewhere; and Q denote a predicate to be applied at each location (x, y). Arrays f and S are assumed to be of the same size. A basic region-growing algorithm based on 8-connectivity may be stated as follows.

See Sections 2.5.2 and 9.5.3 regarding connected components, and Section 9.2.1 regarding erosion.

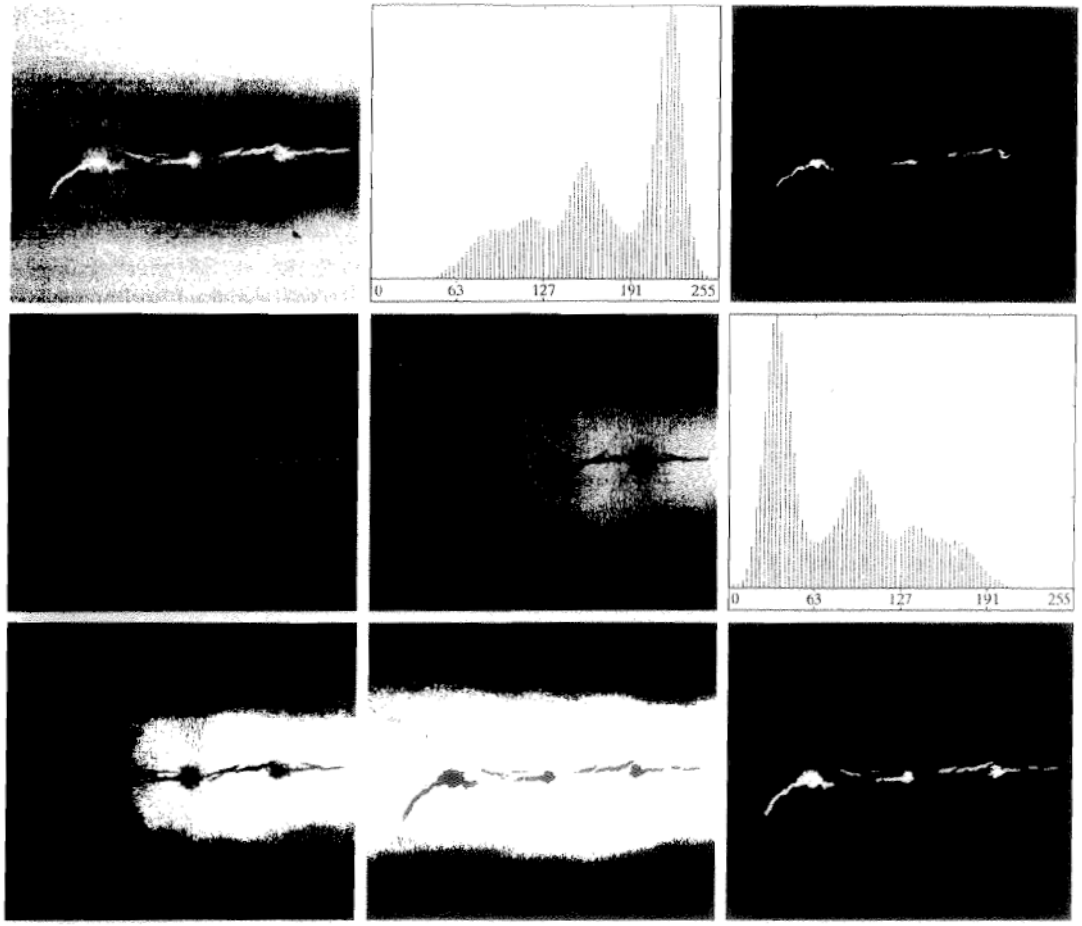
1. Find all connected components in $S(x, y)$ and erode each connected component to one pixel; label all such pixels found as 1. All other pixels in S are labeled 0.
2. Form an image f_Q such that, at a pair of coordinates (x, y), let $f_Q(x, y) = 1$ if the input image satisfies the given predicate, Q, at those coordinates; otherwise, let $f_Q(x, y) = 0$.
3. Let g be an image formed by appending to each seed point in S all the 1-valued points in f_Q that are 8-connected to that seed point.
4. Label each connected component in g with a different region label (e.g., $1, 2, 3, \ldots$). This is the segmented image obtained by region growing.

We illustrate the mechanics of this algorithm by an example.

EXAMPLE 10.23: Segmentation by region growing.

■ Figure 10.51(a) shows an 8-bit X-ray image of a weld (the horizontal dark region) containing several cracks and porosities (the bright regions running horizontally through the center of the image). We illustrate the use of region growing by segmenting the defective weld regions. These regions could be used in applications such as weld inspection, for inclusion in a database of historical studies, or for controlling an automated welding system.

The first order of business is to determine the seed points. From the physics of the problem, we know that cracks and porosities will attenuate X-rays considerably less than solid welds, so we expect the regions containing these types of defects to be significantly brighter than other parts of the X-ray image. We can extract the seed points by thresholding the original image, using a threshold set at a high percentile. Figure 10.51(b) shows the histogram of the image



a b c
d e f
g h i

FIGURE 10.51 (a) X-ray image of a defective weld. (b) Histogram. (c) Initial seed image. (d) Final seed image (the points were enlarged for clarity). (e) Absolute value of the difference between (a) and (c). (f) Histogram of (e). (g) Difference image thresholded using dual thresholds. (h) Difference image thresholded with the smallest of the dual thresholds. (i) Segmentation result obtained by region growing. (Original image courtesy of X-TEK Systems, Ltd.)

and Fig. 10.51(c) shows the thresholded result obtained with a threshold equal to the 99.9 percentile of intensity values in the image, which in this case was 254 (see Section 10.3.5 regarding percentiles). Figure 10.51(d) shows the result of morphologically eroding each connected component in Fig. 10.51(c) to a single point.

Next, we have to specify a predicate. In this example, we are interested in appending to each seed all the pixels that (a) are 8-connected to that seed and

(b) are "similar" to it. Using intensity differences as a measure of similarity, our predicate applied at each location (x, y) is

$$Q = \begin{cases} \text{TRUE} & \text{if the absolute difference of the intensities between the seed and the pixel at } (x, y) \text{ is } \leq T \\ \text{FALSE} & \text{otherwise} \end{cases}$$

where T is a specified threshold. Although this predicate is based on intensity differences and uses a single threshold, we could specify more complex schemes in which a different threshold is applied to each pixel, and properties other than differences are used. In this case, the preceding predicate is sufficient to solve the problem, as the rest of this example shows.

From the previous paragraph, we know that the smallest seed value is 255 because the image was thresholded with a threshold of 254. Figure 10.51(e) shows the absolute value of the difference between the images in Figs. 10.51(a) and (c). The image in Fig. 10.51(e) contains all the differences needed to compute the predicate at each location (x, y). Figure 10.51(f) shows the corresponding histogram. We need a threshold to use in the predicate to establish similarity. The histogram has three principal modes, so we can start by applying to the difference image the dual thresholding technique discussed in Section 10.3.6. The resulting two thresholds in this case were $T_1 = 68$ and $T_2 = 126$, which we see correspond closely to the valleys of the histogram. (As a brief digression, we segmented the image using these two thresholds. The result in Fig. 10.51(g) shows that the problem of segmenting the defects cannot be solved using dual thresholds, even though the thresholds are in the main valleys.)

Figure 10.51(h) shows the result of thresholding the difference image with only T_1. The black points are the pixels for which the predicate was TRUE; the others failed the predicate. The important result here is that the points in the good regions of the weld failed the predicate, so they will not be included in the final result. The points in the outer region will be considered by the region-growing algorithm as candidates. However, Step 3 will reject the outer points, because they are not 8-connected to the seeds. In fact, as Fig. 10.51(i) shows, this step resulted in the correct segmentation, indicating that the use of connectivity was a fundamental requirement in this case. Finally, note that in Step 4 we used the same value for all the regions found by the algorithm. In this case, it was visually preferable to do so. ■

10.4.2 Region Splitting and Merging

The procedure discussed in the last section grows regions from a set of seed points. An alternative is to subdivide an image initially into a set of arbitrary, disjoint regions and then merge and/or split the regions in an attempt to satisfy the conditions of segmentation stated in Section 10.1. The basics of splitting and merging are discussed next.

Let R represent the entire image region and select a predicate Q. One approach for segmenting R is to subdivide it successively into smaller and smaller quadrant regions so that, for any region R_i, $Q(R_i)$ = TRUE. We start with the entire region. If $Q(R)$ = FALSE, we divide the image into quadrants. If Q is FALSE for any quadrant, we subdivide that quadrant into subquadrants, and so on. This particular splitting technique has a convenient representation in the form of so-called *quadtrees*, that is, trees in which each node has exactly four descendants, as Fig. 10.52 shows (the images corresponding to the nodes of a quadtree sometimes are called *quadregions* or *quadimages*). Note that the root of the tree corresponds to the entire image and that each node corresponds to the subdivision of a node into four descendant nodes. In this case, only R_4 was subdivided further.

If only splitting is used, the final partition normally contains adjacent regions with identical properties. This drawback can be remedied by allowing merging as well as splitting. Satisfying the constraints of segmentation outlined in Section 10.1 requires merging only adjacent regions whose combined pixels satisfy the predicate Q. That is, two adjacent regions R_j and R_k are merged only if $Q(R_j \cup R_k)$ = TRUE.

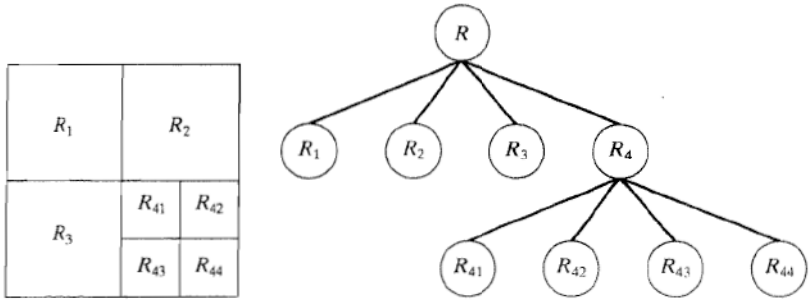
See Section 2.5.2 regarding region adjacency.

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

1. Split into four disjoint quadrants any region R_i for which $Q(R_i)$ = FALSE.
2. When no further splitting is possible, merge any adjacent regions R_j and R_k for which $Q(R_j \cup R_k)$ = TRUE.
3. Stop when no further merging is possible.

It is customary to specify a minimum quadregion size beyond which no further splitting is carried out.

Numerous variations of the preceding basic theme are possible. For example, a significant simplification results if in Step 2 we allow merging of any two adjacent regions R_i and R_j if each one satisfies the predicate individually. This results in a much simpler (and faster) algorithm, because testing of the predicate is limited to individual quadregions. As the following example shows, this simplification is still capable of yielding good segmentation results.



a b

FIGURE 10.52 (a) Partitioned image. (b) Corresponding quadtree. R represents the entire image region.

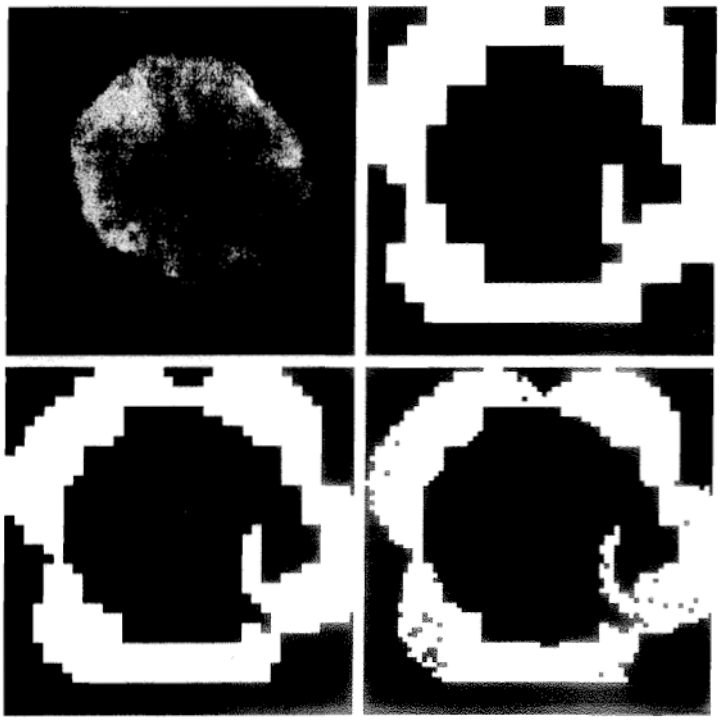
EXAMPLE 10.24: Segmentation by region splitting and merging.

■ Figure 10.53(a) shows a 566 × 566 X-ray band image of the Cygnus Loop. The objective of this example is to segment out of the image the "ring" of less dense matter surrounding the dense center. The region of interest has some obvious characteristics that should help in its segmentation. First, we note that the data in this region has a random nature, indicating that its standard deviation should be greater than the standard deviation of the background (which is near 0) and of the large central region, which is fairly smooth. Similarly, the mean value (average intensity) of a region containing data from the outer ring should be greater than the mean of the darker background and less than the mean of the large, lighter central region. Thus, we should be able to segment the region of interest using the following predicate:

$$Q = \begin{cases} \text{TRUE} & \text{if } \sigma > a \quad \text{AND} \quad 0 < m < b \\ \text{FALSE} & \text{otherwise} \end{cases}$$

where m and σ are the mean and standard deviation of the pixels in a quadregion, and a and b are constants.

Analysis of several regions in the outer area of interest revealed that the mean intensity of pixels in those regions did not exceed 125 and the standard deviation was always greater than 10. Figures 10.53(b) through (d) show the results obtained using these values for a and b, and varying the minimum size allowed for the quadregions from 32 to 8. The pixels in a quadregion whose



a b
c d

FIGURE 10.53 (a) Image of the Cygnus Loop supernova, taken in the X-ray band by NASA's Hubble Telescope. (b)–(d) Results of limiting the smallest allowed quadregion to sizes of 32 × 32, 16 × 16, and 8 × 8 pixels, respectively. (Original image courtesy of NASA.)

pixels satisfied the predicate were set to white; all others in that region were set to black. The best result in terms of capturing the shape of the outer region was obtained using quadregions of size 16 × 16. The black squares in Fig. 10.53(d) are quadregions of size 8 × 8 whose pixels did not satisfied the predicate. Using smaller quadregions would result in increasing numbers of such black regions. Using regions larger than the one illustrated here results in a more "block-like" segmentation. Note that in all cases the segmented regions (white pixels) completely separate the inner, smoother region from the background. Thus, the segmentation effectively partitioned the image into three distinct areas that correspond to the three principal features in the image: background, dense, and sparse regions. Using any of the white regions in Fig. 10.53 as a mask would make it a relatively simple task to extract these regions from the original image (Problem 10.40). As in Example 10.23, these results could not have been obtained using edge- or threshold-based segmentation. ■

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

10.5 Segmentation Using Morphological Watersheds

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

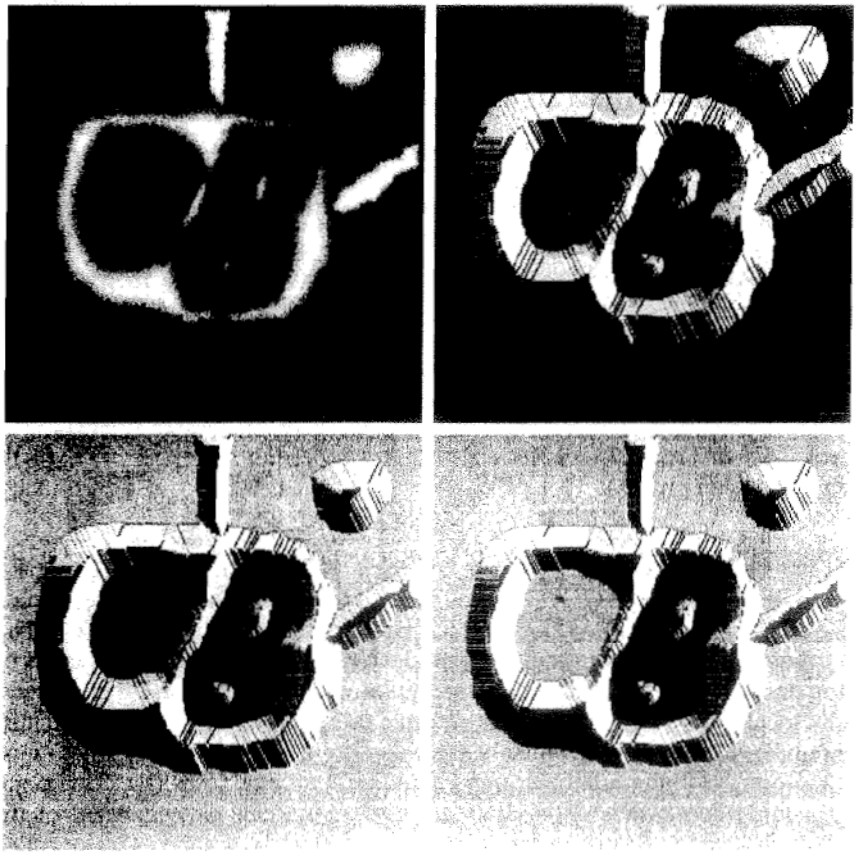
10.5.1 Background

The concept of watersheds is based on visualizing an image in three dimensions: two spatial coordinates versus intensity, as in Fig. 2.18(a). In such a "topographic" interpretation, we consider three types of points: (a) points belonging to a regional minimum; (b) points at which a drop of water, if placed at the location of any of those points, would fall with certainty to a single minimum; and (c) points at which water would be equally likely to fall to more than one such minimum. For a particular regional minimum, the set of points satisfying condition (b) is called the *catchment basin* or *watershed* of that

minimum. The points satisfying condition (c) form crest lines on the topographic surface and are termed *divide lines* or *watershed lines*.

The principal objective of segmentation algorithms based on these concepts is to find the watershed lines. The basic idea is simple, as the following analogy illustrates. Suppose that a hole is punched in each regional minimum and that the entire topography is flooded from below by letting water rise through the holes at a uniform rate. When the rising water in distinct catchment basins is about to merge, a dam is built to prevent the merging. The flooding will eventually reach a stage when only the tops of the dams are visible above the water line. These dam boundaries correspond to the divide lines of the watersheds. Therefore, they are the (connected) boundaries extracted by a watershed segmentation algorithm.

These ideas can be explained further with the aid of Fig. 10.54. Figure 10.54(a) shows a gray-scale image and Fig. 10.54(b) is a topographic view, in which the height of the "mountains" is proportional to intensity values in the input image. For ease of interpretation, the backsides of structures are shaded. This is not to be confused with intensity values; only the general topography of the three-dimensional representation is of interest. In order to prevent the rising water from spilling out through the edges of the image, we imagine the

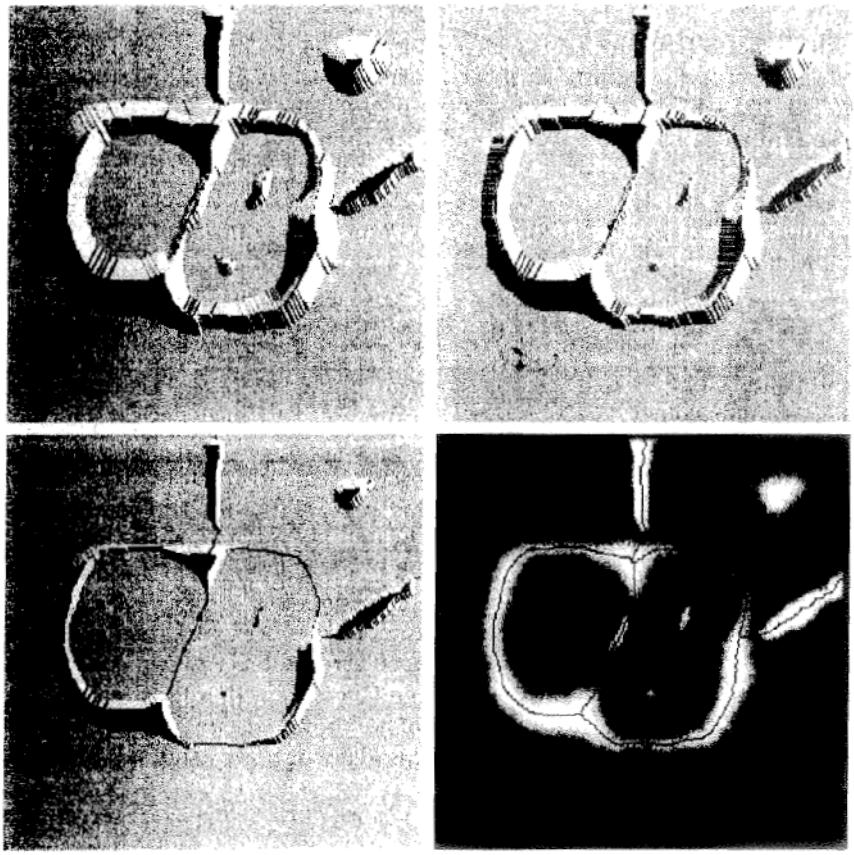


a b
c d

FIGURE 10.54
(a) Original image. (b) Topographic view. (c)–(d) Two stages of flooding.

perimeter of the entire topography (image) being enclosed by dams of height greater than the highest possible mountain, whose value is determined by the highest possible intensity value in the input image.

Suppose that a hole is punched in each regional minimum [shown as dark areas in Fig. 10.54(b)] and that the entire topography is flooded from below by letting water rise through the holes at a uniform rate. Figure 10.54(c) shows the first stage of flooding, where the "water," shown in light gray, has covered only areas that correspond to the very dark background in the image. In Figs. 10.54(d) and (e) we see that the water now has risen into the first and second catchment basins, respectively. As the water continues to rise, it will eventually overflow from one catchment basin into another. The first indication of this is shown in 10.54(f). Here, water from the left basin actually overflowed into the basin on the right and a short "dam" (consisting of single pixels) was built to prevent water from merging at that level of flooding (the details of dam building are discussed in the following section). The effect is more pronounced as water continues to rise, as shown in Fig. 10.54(g). This figure shows a longer dam between the two catchment basins and another dam in the top part of the right basin. The latter dam was built to prevent merging of water from that basin with water from areas corresponding to the background. This process is continued until the maximum



e f
g h

FIGURE 10.54 *(Continued)* (e) Result of further flooding. (f) Beginning of merging of water from two catchment basins (a short dam was built between them). (g) Longer dams. (h) Final watershed (segmentation) lines. (Courtesy of Dr. S. Beucher, CMM/Ecole des Mines de Paris.)

level of flooding (corresponding to the highest intensity value in the image) is reached. The final dams correspond to the watershed lines, which are the desired segmentation result. The result for this example is shown in Fig. 10.54(h) as dark, 1-pixel-thick paths superimposed on the original image. Note the important property that the watershed lines form connected paths, thus giving continuous boundaries between regions.

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

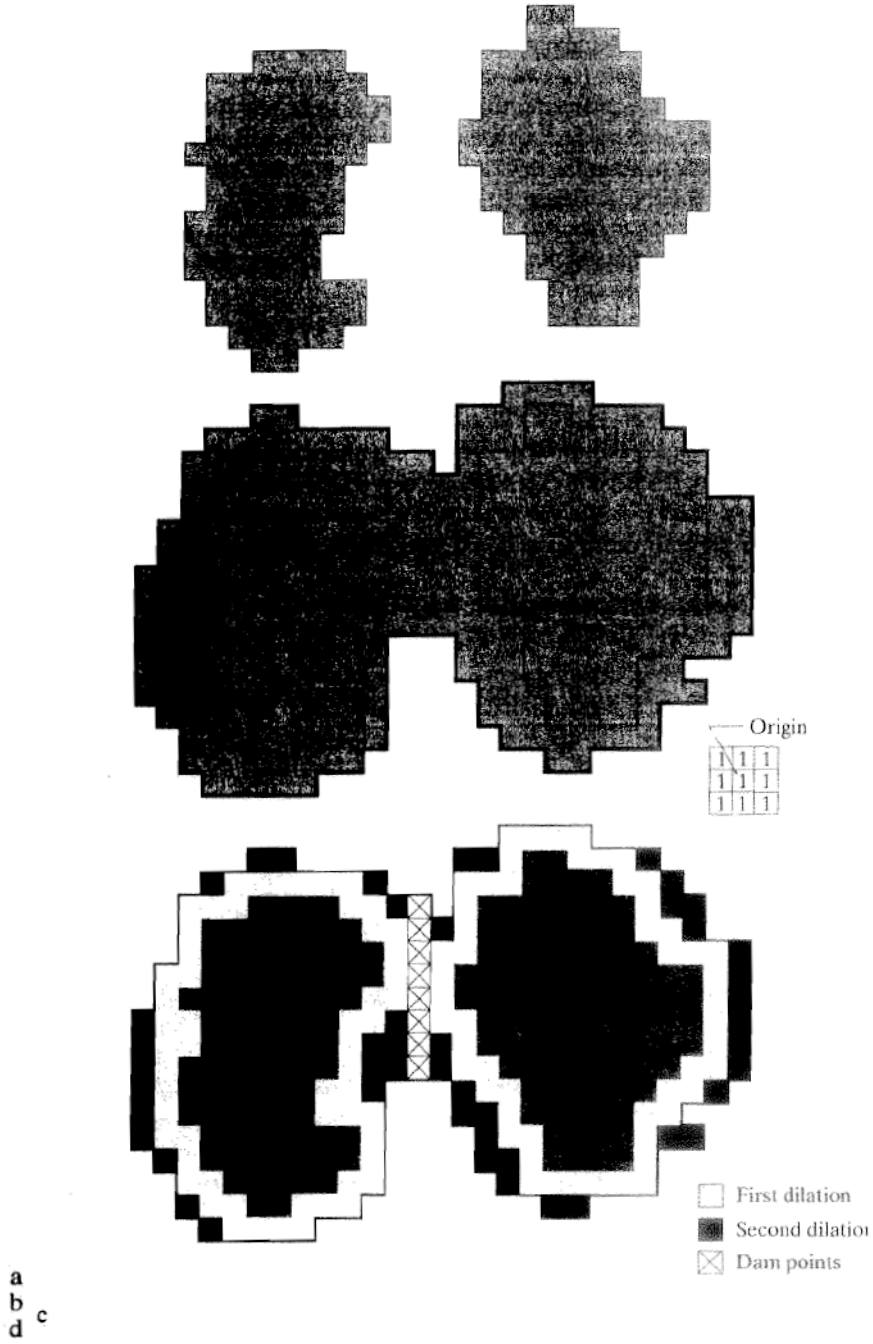
10.5.2 Dam Construction

Before proceeding, let us consider how to construct the dams or watershed lines required by watershed segmentation algorithms. Dam construction is based on binary images, which are members of 2-D integer space Z^2 (see Section 2.4.2). The simplest way to construct dams separating sets of binary points is to use morphological dilation (see Section 9.2.2).

The basics of how to construct dams using dilation are illustrated in Fig. 10.55. Figure 10.55(a) shows portions of two catchment basins at flooding step $n - 1$ and Fig. 10.55(b) shows the result at the next flooding step, n. The water has spilled from one basin to the other and, therefore, a dam must be built to keep this from happening. In order to be consistent with notation to be introduced shortly, let M_1 and M_2 denote the sets of coordinates of points in two regional minima. Then let the set of coordinates of points in the *catchment basin* associated with these two minima at stage $n - 1$ of flooding be denoted by $C_{n-1}(M_1)$ and $C_{n-1}(M_2)$, respectively. These are the two gray regions in Fig. 10.55(a).

Let $C[n - 1]$ denote the union of these two sets. There are two connected components in Fig. 10.55(a) (see Section 2.5.2 regarding connected components) and only one connected component in Fig. 10.55(b). This connected component encompasses the earlier two components, shown dashed. The fact that two connected components have become a *single* component indicates that water between the two catchment basins has merged at flooding step n. Let this connected component be denoted q. Note that the two components from step $n - 1$ can be extracted from q by performing the simple AND operation $q \cap C[n - 1]$. We note also that all points belonging to an individual catchment basin form a single connected component.

Suppose that each of the connected components in Fig. 10.55(a) is dilated by the structuring element shown in Fig. 10.55(c), subject to two conditions: (1) The dilation has to be constrained to q (this means that the center of the structuring element can be located only at points in q during dilation), and (2) the dilation cannot be performed on points that would cause the sets being dilated to merge (become a single connected component). Figure 10.55(d) shows that a first dilation pass (in light gray) expanded the boundary of each original connected component. Note that condition (1) was satisfied by every point



a
b
d c

FIGURE 10.55 (a) Two partially flooded catchment basins at stage $n - 1$ of flooding. (b) Flooding at stage n, showing that water has spilled between basins. (c) Structuring element used for dilation. (d) Result of dilation and dam construction.

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

In the second dilation (shown in black), several points failed condition (1) while meeting condition (2), resulting in the broken perimeter shown in the figure. It also is evident that the only points in q that satisfy the two conditions under consideration describe the 1-pixel-thick connected path shown crosshatched in Fig. 10.55(d). This path constitutes the desired separating dam at stage n of flooding. Construction of the dam at this level of flooding is completed by setting all the points in the path just determined to a value greater than the maximum intensity value of the image. The height of all dams is generally set at 1 plus the maximum allowed value in the image. This will prevent water from crossing over the part of the completed dam as the level of flooding is increased. It is important to note that dams built by this procedure, which are the desired segmentation boundaries, are connected components. In other words, this method eliminates the problems of broken segmentation lines.

Although the procedure just described is based on a simple example, the method used for more complex situations is exactly the same, including the use of the 3×3 symmetric structuring element shown in Fig. 10.55(c).

10.5.3 Watershed Segmentation Algorithm

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

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

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

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

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

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

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

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

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

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

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

It can be shown (Problem 10.41) that the elements in both $C_n(M_i)$ and $T[n]$ are never replaced during execution of the algorithm, and that the number of elements in these two sets either increases or remains the same as n increases. Thus, it follows that $C[n - 1]$ is a subset of $C[n]$. According to Eqs. (10.5-2) and (10.5-3), $C[n]$ is a subset of $T[n]$, so it follows that $C[n - 1]$ is a subset of $T[n]$. From this we have the important result that each connected component of $C[n - 1]$ is contained in exactly one connected component of $T[n]$.

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

1. $q \cap C[n - 1]$ is empty.
2. $q \cap C[n - 1]$ contains one connected component of $C[n - 1]$.
3. $q \cap C[n - 1]$ contains more than one connected component of $C[n - 1]$.

Construction of $C[n]$ from $C[n - 1]$ depends on which of these three conditions holds. Condition 1 occurs when a new minimum is encountered, in which case connected component q is incorporated into $C[n - 1]$ to form $C[n]$. Condition 2 occurs when q lies within the catchment basin of some regional minimum, in which case q is incorporated into $C[n - 1]$ to form $C[n]$. Condition 3 occurs when all, or part, of a ridge separating two or more catchment basins is encountered. Further flooding would cause the water level in these catchment basins to merge. Thus a dam (or dams if more than two catchment basins are involved) must be built within q to prevent overflow between the catchment basins. As explained in the previous section, a one-pixel-thick dam can be constructed when needed by dilating $q \cap C[n - 1]$ with a 3×3 structuring element of 1s, and constraining the dilation to q.

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

a b
c d

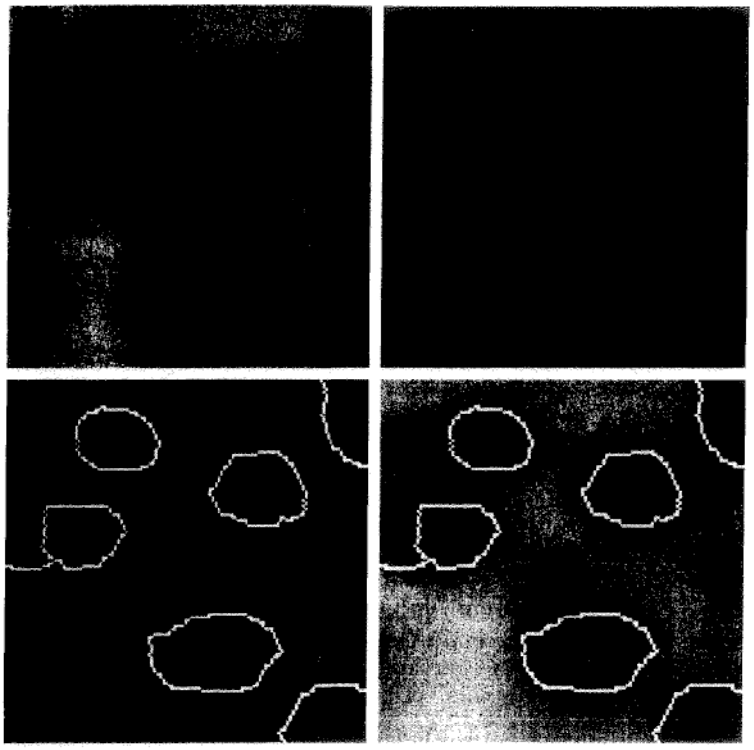


FIGURE 10.56
(a) Image of blobs. (b) Image gradient. (c) Watershed lines. (d) Watershed lines superimposed on original image. (Courtesy of Dr. S. Beucher, CMM/Ecole des Mines de Paris.)

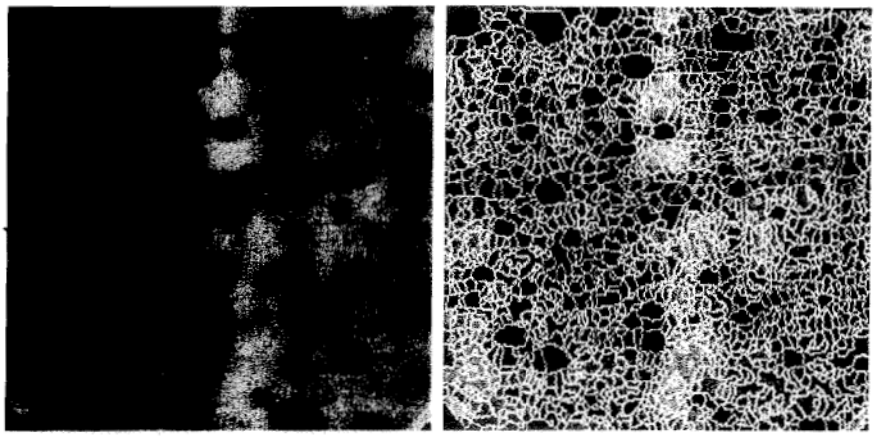
EXAMPLE 10.25: Illustration of the watershed segmentation algorithm.

■ Consider the image and its gradient in Figs. 10.56(a) and (b), respectively. Application of the watershed algorithm just described yielded the watershed lines (white paths) of the gradient image in Fig. 10.56(c). These segmentation boundaries are shown superimposed on the original image in Fig. 10.56(d). As noted at the beginning of this section, the segmentation boundaries have the important property of being connected paths. ■

10.5.4 The Use of Markers

Direct application of the watershed segmentation algorithm in the form discussed in the previous section generally leads to *oversegmentation* due to noise and other local irregularities of the gradient. As Fig. 10.57 shows, oversegmentation can be serious enough to render the result of the algorithm virtually useless. In this case, this means a large number of segmented regions. A practical solution to this problem is to limit the number of allowable regions by incorporating a preprocessing stage designed to bring additional knowledge into the segmentation procedure.

An approach used to control oversegmentation is based on the concept of markers. A *marker* is a connected component belonging to an image. We have *internal* markers, associated with objects of interest, and *external* markers, associated with the background. A procedure for marker selection typically will consist of two principal steps: (1) preprocessing; and (2) definition of a set of criteria that markers must satisfy. To illustrate, consider Fig. 10.57(a) again.

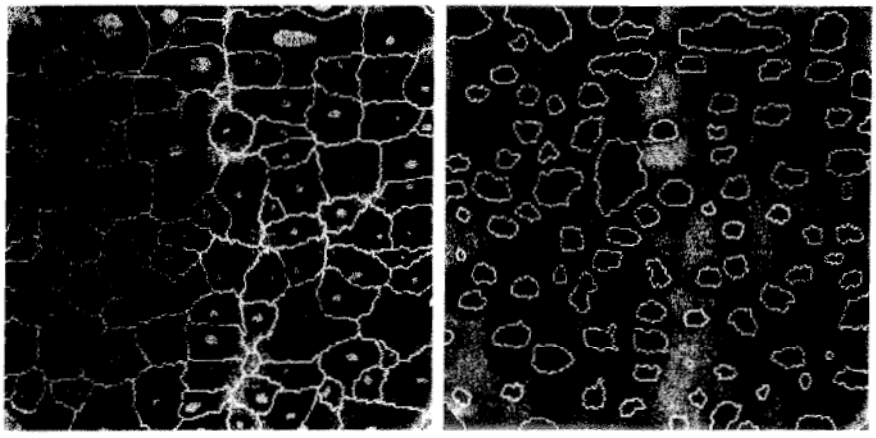


a b

FIGURE 10.57 (a) Electrophoresis image. (b) Result of applying the watershed segmentation algorithm to the gradient image. Oversegmentation is evident. (Courtesy of Dr. S. Beucher, CMM/Ecole des Mines de Paris.)

Part of the problem that led to the oversegmented result in Fig. 10.57(b) is the large number of potential minima. Because of their size, many of these minima are irrelevant detail. As has been pointed out several times in earlier discussions, an effective method for minimizing the effect of small spatial detail is to filter the image with a smoothing filter. This is an appropriate preprocessing scheme in this particular case.

Suppose that we define an *internal marker* as (1) a region that is surrounded by points of higher "altitude"; (2) such that the points in the region form a connected component; and (3) in which all the points in the connected component have the same intensity value. After the image was smoothed, the internal markers resulting from this definition are shown as light gray, bloblike regions in Fig. 10.58(a). Next, the watershed algorithm was applied to the



a b

FIGURE 10.58 (a) Image showing internal markers (light gray regions) and external markers (watershed lines). (b) Result of segmentation. Note the improvement over Fig. 10.47(b). (Courtesy of Dr. S. Beucher, CMM/Ecole des Mines de Paris.)

smoothed image, under the restriction that these internal markers be the *only* allowed regional minima. Figure 10.58(a) shows the resulting watershed lines. These watershed lines are defined as the *external markers*. Note that the points along the watershed line pass along the highest points between neighboring markers.

The external markers in Fig. 10.58(a) effectively partition the image into regions, with each region containing a single internal marker and part of the background. The problem is thus reduced to partitioning each of these regions into two: a single object and its background. We can bring to bear on this simplified problem many of the segmentation techniques discussed earlier in this chapter. Another approach is simply to apply the watershed segmentation algorithm to each individual region. In other words, we simply take the gradient of the smoothed image [as in Fig. 10.56(b)] and then restrict the algorithm to operate on a single watershed that contains the marker in that particular region. The result obtained using this approach is shown in 10.58(b). The improvement over the image in 10.57(b) is evident.

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

10.6 The Use of Motion in Segmentation

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

10.6.1 Spatial Techniques

Basic approach

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

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

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

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

In dynamic image processing, all pixels in $d_{ij}(x, y)$ with value 1 are considered the result of object motion. This approach is applicable only if the two images are registered spatially and if the illumination is relatively constant within the bounds established by T. In practice, 1-valued entries in $d_{ij}(x, y)$ may arise as a result of noise. Typically, these entries are isolated points in the difference image, and a simple approach to their removal is to form 4- or 8-connected regions of 1s in $d_{ij}(x, y)$ and then ignore any region that has less than a predetermined number of elements. Although it may result in ignoring small and/or slow-moving objects, this approach improves the chances that the remaining entries in the difference image actually are the result of motion.

Accumulative differences

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

Consider the following three types of accumulative difference images: *absolute*, *positive*, and *negative* ADIs. Assuming that the intensity values of the moving objects are larger than the background, these three types of ADIs are defined as follows. Let $R(x, y)$ denote the reference image and, to simplify the notation, let k denote t_k, so that $f(x, y, k) = f(x, y, t_k)$. We assume that $R(x, y) = f(x, y, 1)$. Then, for any $k > 1$, and keeping in mind that the values of the ADIs are *counts*, we define the following for all relevant values of (x, y):

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

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

and

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

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

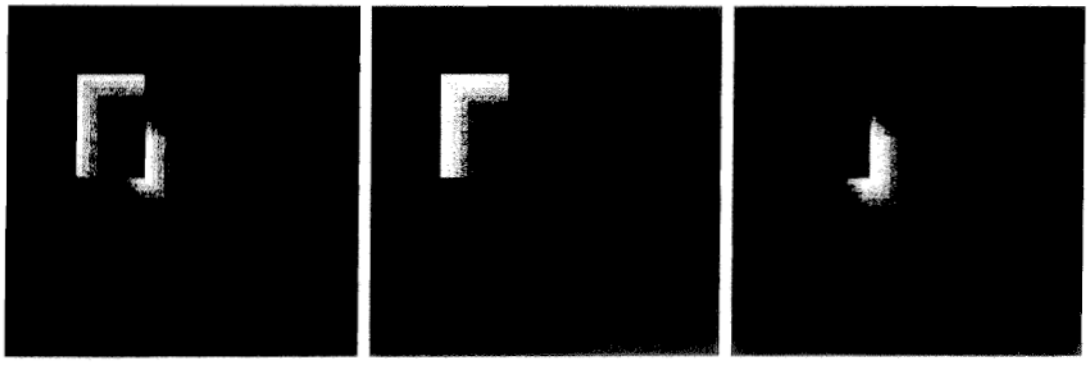
It is understood that these ADIs start out with all zero values (counts). Note also that the ADIs are of the same size as the images in the sequence. Finally, we note that the order of the inequalities and signs of the thresholds in Eqs. (10.6-3) and (10.6-4) are reversed if the intensity values of the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

EXAMPLE 10.26: Computation of the absolute, positive, and negative accumulative difference images.

Figure 10.59 shows the three ADIs displayed as intensity images for a rectangular object of dimension 75×50 pixels that is moving in a southeasterly direction at a speed of $5\sqrt{2}$ pixels per frame. The images are of size 256×256 pixels. We note the following: (1) The nonzero area of the positive ADI is equal to the size of the moving object. (2) The location of the positive ADI corresponds to the location of the moving object in the reference frame. (3) The number of counts in the positive ADI stops increasing when the moving object is displaced completely with respect to the same object in the reference frame. (4) The absolute ADI contains the regions of the positive and negative ADI. (5) The direction and speed of the moving object can be determined from the entries in the absolute and negative ADIs. ❋

Establishing a reference image

A key to the success of the techniques discussed in the preceding two sections is having a reference image against which subsequent comparisons can be



a b c

FIGURE 10.59 ADIs of a rectangular object moving in a southeasterly direction. (a) Absolute ADI. (b) Positive ADI. (c) Negative ADI.

made. The difference between two images in a dynamic imaging problem has the tendency to cancel all stationary components, leaving only image elements that correspond to noise and to the moving objects.

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

EXAMPLE 10.27: Building a reference image.

■ Figures 10.60(a) and (b) show two image frames of a traffic intersection. The first image is considered the reference, and the second depicts the same scene some time later. The objective is to remove the principal moving objects in the reference image in order to create a static image. Although there are other smaller moving objects, the principal moving feature is the automobile at the intersection moving from left to right. For illustrative purposes we focus on this object. By monitoring the changes in the positive ADI, it is possible to determine the initial position of a moving object, as explained previously. Once the area occupied by this object is identified, the object can be removed from the image by subtraction. By looking at the frame in the sequence at which the positive ADI stopped changing, we can copy from this image the area previously occupied by the moving object in the initial frame. This area then is pasted onto the image from which the object was cut out, thus restoring the background of that area. If this is done for all moving objects, the result is a reference image with only static components against which we can compare subsequent frames for motion detection. The result of removing the eastbound moving vehicle in this case is shown in Fig. 10.60(c). ■

a b c

FIGURE 10.60 Building a static reference image. (a) and (b) Two frames in a sequence. (c) Eastbound automobile subtracted from (a) and the background restored from the corresponding area in (b). (Jain and Jain.)

10.6.2 Frequency Domain Techniques

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

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

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

for $t = 0, 1, \ldots, K - 1$. In other words, this procedure yields a complex sinusoid with frequency a_1. If the object were moving V_1 pixels (in the x-direction) between frames, the sinusoid would have frequency $V_1 a_1$. Because t varies between 0 and $K - 1$ in integer increments, restricting a_1 to integer values causes the discrete Fourier transform of the complex sinusoid to have two peaks—one located at frequency $V_1 a_1$ and the other at $K - V_1 a_1$. This latter peak is the result of symmetry in the discrete Fourier transform, as discussed in Section 4.6.4, and may be ignored. Thus a peak search in the Fourier spectrum yields $V_1 a_1$. Division of this quantity by a_1 yields V_1, which is the velocity component in the x-direction, as the frame rate is assumed to be known. A similar argument would yield V_2, the component of velocity in the y-direction.

A sequence of frames in which no motion takes place produces identical exponential terms, whose Fourier transform would consist of a single peak at a frequency of 0 (a single dc term). Therefore, because the operations discussed so far are linear, the general case involving one or more moving objects in an arbitrary static background would have a Fourier transform with a peak at dc corresponding to static image components and peaks at locations proportional to the velocities of the objects.

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

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

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

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

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

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

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

and

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

In practice, computation of these transforms is carried out using an FFT algorithm, as discussed in Section 4.11.

The frequency-velocity relationship is

$$u_1 = a_1 V_1 \tag{10.6-10}$$

and

$$u_2 = a_2 V_2 \tag{10.6-11}$$

In this formulation the unit of velocity is in pixels per total frame time. For example, $V_1 = 10$ is interpreted as a motion of 10 pixels in K frames. For frames that are taken uniformly, the actual physical speed depends on the frame rate and the distance between pixels. Thus if $V_1 = 10$, $K = 30$, the frame rate is two images per second, and the distance between pixels is 0.5 m, then the actual physical speed in the x-direction is

$$\begin{aligned} V_1 &= (10 \text{ pixels})(0.5 \text{ m/pixel})(2 \text{ frames/s})/(30 \text{ frames}) \\ &= 1/3 \text{ m/s} \end{aligned}$$

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

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

and

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

Because g_x is sinusoidal, it can be shown (Problem 10.47) that S_{1x} and S_{2x} will have the same sign at an arbitrary point in time, n, if the velocity component V_1 is positive. Conversely, opposite signs in S_{1x} and S_{2x} indicate a negative component. If either S_{1x} or S_{2x} is zero, we consider the next closest point in time, $t = n \pm \Delta t$. Similar comments apply to computing the sign of V_2.

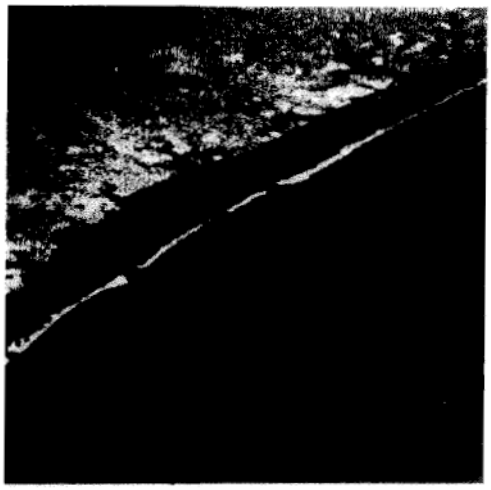


FIGURE 10.61 LANDSAT frame. (Cowart, Snyder, and Ruedger.)

EXAMPLE 10.28: Detection of a small moving object via the frequency domain.

■ Figures 10.61 through 10.64 illustrate the effectiveness of the approach just derived. Figure 10.61 shows one of a 32-frame sequence of LANDSAT images generated by adding white noise to a reference image. The sequence contains a superimposed target moving at 0.5 pixel per frame in the x-direction and 1 pixel per frame in the y-direction. The target, shown circled in Fig. 10.62, has a Gaussian intensity distribution spread over a small (9-pixel) area and is not easily discernible by eye. Figures 10.63 and 10.64 show the results of computing Eqs. (10.6-8) and (10.6-9) with $a_1 = 6$ and $a_2 = 4$, respectively. The peak at $u_1 = 3$ in Fig. 10.63 yields $V_1 = 0.5$ from Eq. (10.6-10). Similarly, the peak at $u_2 = 4$ in Fig. 10.64 yields $V_2 = 1.0$ from Eq. (10.6-11). ■

Guidelines for the selection of a_1 and a_2 can be explained with the aid of Figs. 10.63 and 10.64. For instance, suppose that we had used $a_2 = 15$ instead of $a_2 = 4$. In that case the peaks in Fig. 10.64 would now be at $u_2 = 15$ and 17 because $V_2 = 1.0$, which would be a seriously aliased result. As discussed in Section 4.5.4, aliasing is caused by undersampling (too few frames in the present discussion, as the range of u is determined by K). Because $u = aV$, one possibility is to select

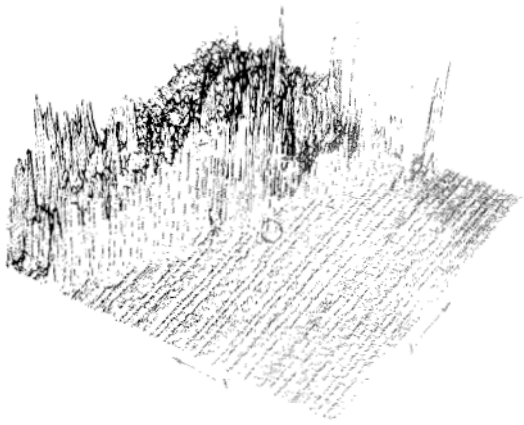


FIGURE 10.62 Intensity plot of the image in Fig. 10.61, with the target circled. (Rajala, Riddle, and Snyder.)

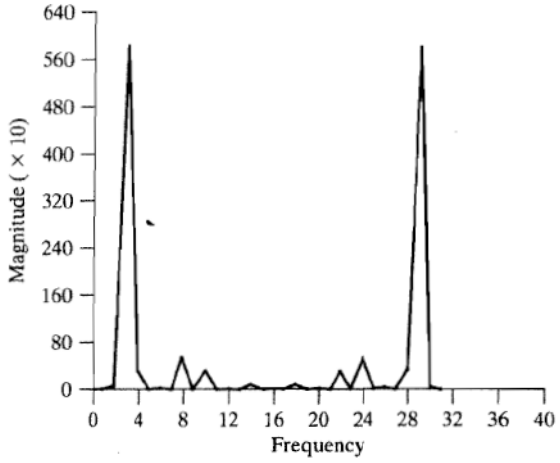


FIGURE 10.63
Spectrum of Eq. (10.6-8) showing a peak at $u_1 = 3$. (Rajala, Riddle, and Snyder.)

a as the integer closest to $a = u_{max}/V_{max}$, where u_{max} is the aliasing frequency limitation established by K and V_{max} is the maximum expected object velocity.

Summary

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

References and Further Reading

Because of its central role in autonomous image processing, segmentation is a topic covered in most books dealing with image processing, image analysis, and computer vision. The following books provide complementary and/or supplementary reading for our coverage of this topic: Umbaugh [2005]; Davies [2005]; Gonzalez, Woods, and Eddins [2004]; Shapiro and Stockman [2001]; Sonka et al. [1999]; and Petrou and Bosdogianni [1999].

Work dealing with the use of masks to detect intensity discontinuities (Section 10.2) has a long history. Numerous masks have been proposed over the years: Roberts [1965], Prewitt [1970], Kirsh [1971], Robinson [1976], Frei and Chen [1977], and Canny [1986]. A review article by Fram and Deutsch [1975] contains numerous masks and an evaluation of

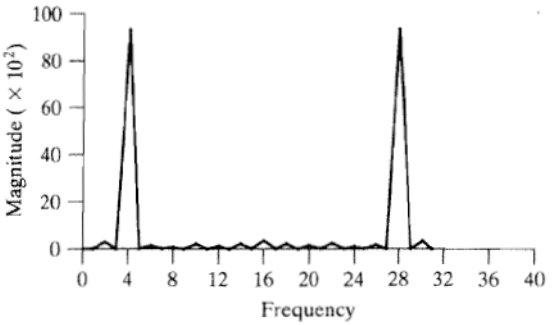


FIGURE 10.64
Spectrum of Eq. (10.6-9) showing a peak at $u_2 = 4$. (Rajala, Riddle, and Snyder.)

their performance. The issue of mask performance, especially for edge detection, still is an area of considerable interest, as exemplified by Qian and Huang [1996], Wang et al. [1996], Heath et al. [1997, 1998], and Ando [2000]. Edge detection on color images has been increasing in popularity for a number of multisensing applications. See, for example, Salinas, Abidi, and Gonzalez [1996]; Zugaj and Lattuati [1998]; Mirmehdi and Petrou [2000]; and Plataniotis and Venetsanopoulos [2000]. The interplay between image characteristics and mask performance also is a topic of current interest, as exemplified by Ziou [2001]. Our presentation of the zero-crossing properties of the Laplacian is based on a paper by Marr and Hildredth [1980] and on the book by Marr [1982]. See also a paper by Clark [1989] on authenticating edges produced by zero-crossing algorithms. (Corrections of parts of the Clark paper are given by Piech [1990].) As mentioned in Section 10.2, zero crossing via the Laplacian of a Gaussian is an important approach whose relative performance is still an active topic of research (Gunn [1998, 1999]). As the name implies, the Canny edge detector discussed in Section 10.2.6 is due to Canny [1986]. For an example of work on this topic twenty years later, see Zhang and Rockett [2006].

The Hough transform (Hough [1962]) is a practical method for global pixel linking and curve detection. Numerous generalizations to the basic transform discussed in this chapter have been proposed over the years. For example, Lo and Tsai [1995] discuss an approach for detecting thick lines, Guil et al. [1995, 1997] deal with fast implementations of the Hough transform and detection of primitive curves, Daul at al. [1998] discuss further generalizations for detecting elliptical arcs, and Shapiro [1996] deals with implementation of the Hough transform for gray-scale images.

As mentioned at the beginning of Section 10.3, thresholding techniques enjoy a significant degree of popularity because they are simple to implement. It is not surprising that there is a considerable body of work reported in the literature on this topic. A good appreciation of the extent of this literature can be gained from the review papers by Sahoo et al. [1988] and by Lee et al. [1990]. In addition to the techniques discussed in this chapter, other approaches used to deal with the effects of illumination and reflectance (Section 10.3.1) are illustrated by the work of Perez and Gonzalez [1987], Parker [1991], Murase and Nayar [1994], Bischsel [1998], Drew et al. [1999], and Toro and Funt [2007]. For additional reading on the material in Section 10.3.2, see Jain et al. [1995].

Early work on optimal global thresholding (Section 10.3.3) is exemplified in the classic paper by Chow and Kaneko [1972] (we discuss this method in Section 12.2.2 in the more general context of object recognition). Although it is optimal in theory, applications of this method in intensity thresholding are limited because of the need to estimate probability density functions. The optimum approach we developed in Section 10.3.3, due to Otsu [1979], has gained much more acceptance because it combines excellent performance with simplicity of implementation, requiring only estimation of image histograms. The basic idea of using preprocessing (Sections 10.3.4 and 10.3.5) dates back to an early paper by White and Rohrer [1983]), which combined thresholding, the gradient, and the Laplacian in the solution of a difficult segmentation problem. It is interesting to compare the fundamental similarities in terms of image segmentation capability between the methods discussed in the preceding three articles and work on thresholding done almost twenty years later by Cheriet et al. [1998], Sauvola and Pietikainen [2000]), Liang et al. [2000], and Chan et al. [2000]. For additional reading on multiple thresholding (Section 10.3.6), see Yin and Chen [1997], Liao et al. [2001], and Zahara et al. [2005]. For additional reading on variable thresholding (Section 10.3.7), see Parker [1997]. See also Delon et al. [2007].

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

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

The material in Section 10.6.1 is from Jain, R. [1981]. See also Jain, Kasturi, and Schunck [1995]. The material in Section 10.6.2 is from Rajala, Riddle, and Snyder [1983]. See also the papers by Shariat and Price [1990] and by Cumani et al. [1991]. The books by Sonka et al. [1999], Shapiro and Stockman [2001], Snyder and Qi [2004], and Davies [2005] provide additional reading on motion estimation. See also Alexiadis and Sergiadis [2007].

Problems

★10.1 Prove the validity of Eq. (10.2-2). (*Hint:* Use a Taylor series expansion and keep only the linear terms.)

★10.2 A binary image contains straight lines oriented horizontally, vertically, at 45°, and at −45°. Give a set of 5 × 5 masks that can be used to detect 1-pixel breaks in these lines. Assume that the intensities of the lines and background are 1 and 0, respectively.

10.3 Propose a technique for detecting gaps of length ranging between 2 and K pixels in line segments of a binary image. Assume that the lines are 1 pixel thick. Base your technique on 8-neighbor connectivity analysis, rather than attempting to construct masks for detecting the gaps.

10.4 Refer to Fig. 10.7 in answering the following questions.

★(a) Some of the lines joining the pads and center element in Fig. 10.7(e) are single lines, while others are double lines. Explain why.

(b) Propose a method for eliminating the components in Fig. 10.7(f) that are part of the line oriented at −45°.

10.5 Refer to the edge models in Fig. 10.8.

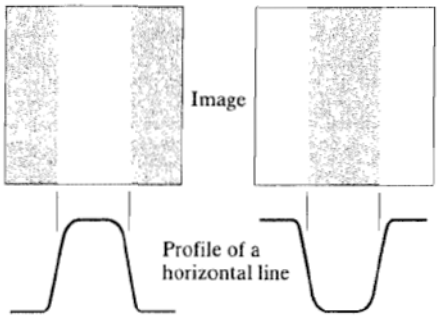
★(a) Suppose that we compute the gradient magnitude of each of these models using the Prewitt operators in Fig. 10.14. Sketch what a horizontal profile through the center of each gradient image would look like.

(b) Sketch a horizontal profile for each corresponding angle image.

(*Note:* Answer this question without generating the gradient and angle images. Simply provide sketches of the profiles that show what you would *expect* the profiles of the magnitude and angle images to look like.)

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

★10.7 Suppose that we had used the edge models shown in the next page, instead of the ramp model in Fig. 10.10. Sketch the gradient and Laplacian of each profile.



10.8 Refer to Fig. 10.14 in answering the following questions.

(a) Assume that the Sobel masks are used to obtain g_x and g_y. Show that in this case the magnitude of the gradient computed using Eqs. (10.2-10) and (10.2-20) give identical results.

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

★10.9 Show that the Sobel and Prewitt masks in Figs. 10.14 and 10.15 give isotropic results only for horizontal and vertical edges and for edges oriented at $\pm 45°$, respectively.

10.10 The results obtained by a single pass through an image of some 2-D masks can be achieved also by two passes using 1-D masks. For example, the same result of using a 3×3 smoothing mask with coefficients $1/9$ can be obtained by a pass of the mask $[1\ 1\ 1]$ through an image. The result of this pass is then followed by a pass of the mask

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

The final result is then scaled by $1/9$. Show that the response of Prewitt masks (Fig. 10.14) can be implemented similarly by one pass of the *differencing* mask $[-1\ 0\ 1]$ (or its vertical counterpart) followed by the *smoothing* mask $[1\ 1\ 1]$ (or its vertical counterpart).

10.11 The so-called compass gradient operators of size 3×3 are designed to measure gradients of edges oriented in eight directions: E, NE, N, NW, W, SW, S, and SE.

★(a) Give the form of these eight operators using coefficients valued 0, 1, -1, -2 or 2.

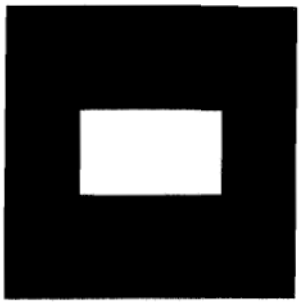
(b) Specify the gradient vector direction of each mask, keeping in mind that the gradient direction is orthogonal to the edge direction.

10.12 The rectangle in the binary image in the next page is of size $m \times n$ pixels.

(a) What would the magnitude of the gradient of this image look like based on using the approximation given in Eq. (10.2-20)? Assume that g_x and g_y are obtained using the Prewitt operators. Show all relevant different pixel values in the gradient image.

(b) Sketch the histogram of edge *directions* computed using Eq. (10.2-11). Be precise in labeling the height of each component of the histogram.

(c) What would the Laplacian of this image look like based on using the approximation in Eq. (10.2-7)? Show all relevant different pixel values in the Laplacian image.



10.13 Suppose that an image $f(x, y)$ is convolved with a mask of size $n \times n$ (with coefficients $1/n^2$) to produce a smoothed image $\bar{f}(x, y)$.

★**(a)** Derive an expression for *edge strength* (edge magnitude) of the smoothed image as a function of mask size. Assume for simplicity that n is odd and that edges are obtained using the partial derivatives

$$\partial\bar{f}/\partial x = \bar{f}(x + 1, y) - \bar{f}(x, y) \quad \text{and} \quad \partial\bar{f}/\partial y = \bar{f}(x, y + 1) - \bar{f}(x, y).$$

(b) Show that the ratio of the maximum edge strength of the smoothed image to the maximum edge strength of the original image is $1/n$. In other words, edge strength is inversely proportional to the size of the smoothing mask.

10.14 With reference to Eq. (10.2-23):

★**(a)** Show that the average value of the Laplacian of a Gaussian operator, $\nabla^2 G(x, y)$, is zero.

(b) Show that the average value of any image convolved with this operator also is zero. (*Hint*: Consider solving this problem in the frequency domain, using the convolution theorem and the fact that the average value of a function is proportional to its Fourier transform evaluated at the origin.)

(c) Would (b) be true in general if we (1) used the mask in Fig. 10.4(a) to compute the Laplacian of a Gaussian lowpass filter using a Laplacian mask of size 3×3, and (2) convolved this result with any image? Explain. (*Hint:* Refer to Problem 3.16.)

10.15 Refer to Fig. 10.22(c).

(a) Explain why the edges form closed contours.

★**(b)** Does the zero-crossing method for finding edge location always result in closed contours? Explain.

10.16 One often finds in the literature a derivation of the Laplacian of a Gaussian (LoG) that starts with the expression

$$G(r) = e^{-r^2/2\sigma^2}$$

where $r^2 = x^2 + y^2$. The LoG is then found by taking the second partial derivative: $\nabla^2 G(r) = \partial^2 G/\partial r^2$. Finally, $x^2 + y^2$ is substituted for r^2 to get the (incorrect) result

$$\nabla^2 G(x, y) = \left[(x^2 + y^2 - \sigma^2)/\sigma^4\right] \exp\left[-(x^2 + y^2)/2\sigma^2\right]$$

Derive this result and explain the reason for the difference between this expression and Eq. (10.2-23).

10.17 **(a)** Derive Eq. (10.2-27).

(b) Let $k = \sigma_1/\sigma_2$ denote the standard deviation ratio discussed in connection with the DoG function. Express Eq. (10.2-27) in terms of k and σ_2.

10.18 In the following, assume that G and f are discrete arrays of size $n \times n$ and $M \times N$, respectively.

★(a) Show that the 2-D convolution of the Gaussian function $G(x, y)$ in Eq. (10.2-21) with an image $f(x, y)$ can be expressed as a 1-D convolution along the rows (columns) of $f(x, y)$ followed by a 1-D convolution along the columns (rows) of the result. (See Section 3.4.2 regarding discrete convolution.)

(b) Derive an expression for the computational advantage of using the 1-D convolution approach in (a) as opposed to implementing the 2-D convolution directly. Assume that $G(x, y)$ is sampled to produce an array of size $n \times n$ and that $f(x, y)$ is of size $M \times N$. The computational advantage is the ratio of the number of multiplications required for 2-D convolution to the number required for 1-D convolution.

★10.19 (a) Show that Steps 1 and 2 of the Marr-Hildreth algorithm can be implemented using four, 1-D convolutions. (*Hints*: Refer to Problem 10.18(a) and express the Laplacian operator as the sum of two partial derivatives, given by Eqs. (10.2-5) and (10.2-6), and implement each derivative using a 1-D mask, as in Problem 10.10.)

(b) Derive an expression for the computational advantage of using the 1-D convolution approach in (a) as opposed to implementing the 2-D convolution directly. Assume that $G(x, y)$ is sampled to produce an array of size $n \times n$ and that $f(x, y)$ is of size $M \times N$. The computational advantage is the ratio of the number of multiplications required for 2-D convolution to the number required for 1-D convolution (see Problem 10.18).

10.20 (a) Formulate Step 1 and the gradient magnitude image computation in Step 2 of the Canny algorithm using 1-D instead of 2-D convolutions.

(b) What is the computational advantage of using the 1-D convolution approach as opposed to implementing a 2-D convolution. Assume that the 2-D Gaussian filter in Step 1 is sampled into an array of size $n \times n$ and the input image is of size $M \times N$. Express the computational advantage as the ratio of the number of multiplications required by each method.

10.21 Refer to the three vertical edge models and corresponding profiles in Fig. 10.8.

★(a) Suppose that we compute the gradient magnitude of each of the three edge models using the Prewitt masks. Sketch the horizontal intensity profiles of the three gradient images.

★(b) Sketch the horizontal intensity profiles of the three Laplacian images, assuming that the Laplacian is computed using the 3×3 mask in Fig. 10.4(a).

★(c) Repeat for an image generated using only the first two steps of the Marr-Hildreth edge detector.

(d) Repeat for the first two steps of the Canny edge detector. You may ignore the angle images.

(e) Sketch the horizontal profile of the angle images for the Canny edge detector.

(*Note*: Answer this question without generating the images. Simply provide sketches of the profiles that show what you would *expect* the profiles of the images to look like.)

10.22 Refer to the Hough transform discussed in Section 10.2.7.

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

★(b) Find the normal representation of the line $y = -3x + 2$.

★**10.23** Refer to the Hough transform discussed in Section 10.2.7.

(a) Explain why the Hough mapping of point 1 in Fig. 10.33(a) is a straight line in Fig. 10.33(b).

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

(c) Explain the reflective adjacency relationship illustrated by, for example, the curve labeled Q in Fig. 10.33(b).

10.24 Show that the number of operations required to implement the accumulator-cell approach discussed in Section 10.2.7 is linear in n, the number of non-background points in the image plane (i.e., the xy-plane).

10.25 An important area of application for image segmentation techniques is in processing images resulting from so-called bubble chamber events. These images arise from experiments in high-energy physics in which a beam of particles of known properties is directed onto a target of known nuclei. A typical event consists of incoming tracks, any one of which, in the event of a collision, branches out into secondary tracks of particles emanating from the point of collision. Propose a segmentation approach for detecting all tracks that contain at least 100 pixels and are angled at any of the following six directions off the horizontal: $\pm 20°$, $\pm 40°$, and $\pm 60°$. The allowed estimation error in any of these six directions is $\pm 5°$. For a track to be valid it must be at least 100 pixels long and not have more than three gaps, any of which cannot exceed 10 pixels. You may assume that the images have been preprocessed so that they are binary and that all tracks are 1 pixel wide, except at the point of collision from which they emanate. Your procedure should be able to differentiate between tracks that have the same direction but different origins. (*Hint*: Base your solution on the Hough transform.)

★**10.26** Restate the basic global thresholding algorithm in Section 10.3.2 so that it uses the histogram of an image instead of the image itself.

★**10.27** Prove that the basic global thresholding algorithm in Section 10.3.2 converges in a finite number of steps. (*Hint*: Use the histogram formulation from Problem 10.26.)

10.28 Give an explanation why the initial threshold in the basic global thresholding algorithm in Section 10.3.2 must be between the minimum and maximum values in the image. (*Hint*: Construct an example that shows the algorithm failing for a threshold value selected outside this range.)

★**10.29** Is the threshold obtained with the basic global thresholding algorithm in Section 10.3.2 independent of the starting point? If your answer is yes, prove it. If your answer is no, give an example.

10.30 You may assume in both of the following cases that the threshold value during iteration is bounded in the open interval $(0, L - 1)$.

★**(a)** Prove that if the histogram of an image is uniform over all possible intensity levels, the basic global thresholding algorithm in Section 10.3.2 converges to the average intensity of the image, $(L - 1)/2$.

(b) Prove that if the histogram of an image is bimodal, with identical modes that are symmetric about their means, then the basic global algorithm will converge to the point halfway between the means of the modes.

10.31 Refer to the thresholding algorithm in Section 10.3.2. Assume that in a given problem the histogram is bimodal with modes that are Gaussian curves of the form $A_1 \exp[-(z - m_1)^2/2\sigma_1^2]$ and $A_2 \exp[-(z - m_2)^2/2\sigma_2^2]$. Assume that $m_1 > m_2$ and that the initial T is between the max and min image intensities. Give conditions (in terms of the parameters of these curves) for the following to be true when the algorithm converges:

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

(b) The threshold is to the right of m_1.

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

If it is not possible for any of these conditions to exist, so state, and give a reason.

★10.32 **(a)** Show how the first line in Eq. (10.3-15) follows from Eqs. (10.3-14), (10.3-10), and (10.3-11).

(b) Show how the second line in Eq. (10.3-15) follows from the first.

10.33 Show that a maximum value for Eq. (10.3-18) always exists for k in the range $0 \leq k \leq L - 1$.

10.34 With reference to Eq. (10.3-20), advance an argument that establishes that $0 \leq \eta(k) \leq 1$, for k in the range $0 \leq k \leq L - 1$, where the minimum is achievable only by images with constant intensity, and the maximum occurs only for 2-valued images with values 0 and $L - 1$.

★10.35 **(a)** Suppose that the intensities of an image $f(x, y)$ are in the range $[0, 1]$ and that a threshold, T, successfully segmented the image into objects and background. Show that the threshold $T' = 1 - T$ will successfully segment the negative of $f(x, y)$ into the same regions. The term *negative* is used here in the sense defined in Section 3.2.1.

(b) The intensity transformation function in (a) that maps an image into its negative is a linear function with negative slope. State the conditions that an arbitrary intensity transformation function must satisfy for the segmentability of the original image with respect to a threshold, T, to be preserved. What would be the value of the threshold after the intensity transformation?

10.36 The objects and background in the image shown have a mean intensity of 180 and 70, respectively, on a $[0, 255]$ scale. The image is corrupted by Gaussian noise with 0 mean and a standard deviation of 10 intensity levels. Propose a thresholding method capable of yielding a correct segmentation rate of 90% or higher. (Recall that 99.7% of the area of a Gaussian curve lies in a $\pm 3\sigma$ interval about the mean, where σ is the standard deviation.)

10.37 Refer to the intensity ramp image in Fig. 10.37(b) and the moving-average algorithm discussed in Section 10.3.7. Assume that the image is of size 400×700 pixels and that its minimum and maximum values are 0 and 1, where 0s are contained only in the first column.

★(a) What would be the result of segmenting this image with the moving-average algorithm using $b = 0$ and an arbitrary value for n. Explain what the image would look like.

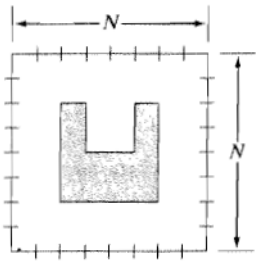
(b) Now reverse the direction of the ramp so that its leftmost value is 1 and the rightmost value is 0 and repeat (a).

(c) Repeat (a) but with $n = 4$ and $b = 1$.

(d) Repeat (a) but with $n = 100$ and $b = 1$.

10.38 Propose a region-growing algorithm to segment the image in Problem 10.36.

★**10.39** Segment the image shown by using the split and merge procedure discussed in Section 10.4.2. Let $Q(R_i)$ = TRUE if all pixels in R_i have the same intensity. Show the quadtree corresponding to your segmentation.



10.40 Consider the region of 1s resulting from the segmentation of the sparse regions in the image of the Cygnus Loop in Example 10.24. Propose a technique for using this region as a mask to isolate the three main components of the image: (1) background, (2) dense inner region, and (3) sparse outer region.

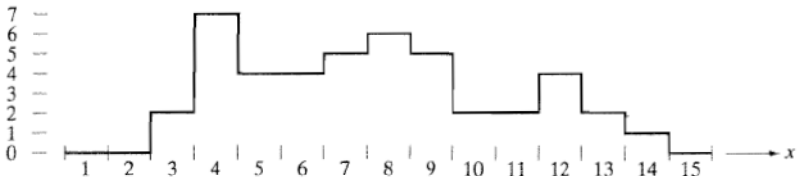
10.41 Refer to the discussion in Section 10.5.3.

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

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

10.42 The boundaries illustrated in Section 10.5, obtained using the watershed segmentation algorithm, form closed loops (for example, see Figs. 10.56 and 10.58). Advance an argument that establishes whether or not closed boundaries always result from application of this algorithm.

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



10.44 What would the negative ADI image in Fig. 10.59(c) look like if we tested against T (instead of testing against $-T$) in Eq. (10.6-4)?

10.45 Are the following statements true or false? Explain the reason for your answer in each.

★**(a)** The nonzero entries in the absolute ADI continue to grow in dimension, provided that the object is moving.

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

(c) The nonzero entries in the negative ADI continue to grow in dimension, provided that the object is moving.

10.46 Suppose that in Example 10.28 motion along the x-axis is set to zero. The object now moves only along the y-axis at 1 pixel per frame for 28 frames and then (instantaneously) reverses direction and moves in exactly the opposite direction for another 28 frames. What would Figs. 10.63 and 10.64 look like under these conditions?

★**10.47** Advance an argument that demonstrates that when the signs of S_{1x} and S_{2x} in Eqs. (10.6-12) and (10.6-13) are the same, the velocity component V_1 is positive.

10.48 An automated pharmaceutical plant uses image processing in measuring the shapes of medication tablets for the purpose of quality control. The segmentation stage of the system is based on Otsu's method. The speed of the inspection lines is so high that a very high rate flash illumination is required to "stop" motion. When new, the illumination lamps project a uniform pattern of light. However, as the lamps age, the illumination pattern deteriorates as a function of time and spatial coordinates according to the equation

$$i(x, y) = A(t) - t^2 e^{-[(x-M)^2+(y-N)^2]}$$

where (M, N) is the center of the viewing area and t is time measured in increments of months. The lamps are experimental and the behavior of $A(t)$ is not fully understood by the manufacturer. All that is known is that, during the life of the lamps, $A(t)$ is always greater than the negative component in the preceding equation because illumination cannot be negative. It has been observed that Otsu's algorithm works well when the lamps are new, and their pattern of illumination is nearly constant over the entire image. However, segmentation performance deteriorates with time. Being experimental, the lamps are exceptionally expensive, so you are employed as a consultant to help solve the problem computationally and thus extend the useful life of the lamps. You are given flexibility to install any special markers or other visual cues near the edges of the viewing area of the imaging cameras. Propose a solution in sufficient detail that the engineering plant manager can understand your approach. (*Hint*: Review the image model discussed in Section 2.3.4 and consider using a small target of known reflectivity.)

10.49 The speed of a bullet in flight is to be estimated by using high-speed imaging techniques. The method of choice involves the use of a TV camera and flash that exposes the scene for K s. The bullet is 3 cm long, 1 cm wide, and its range of speed is 700 ± 200 m/s. The camera optics produce an image in which the bullet occupies 10% of the horizontal resolution of a 256 × 256 digital image.

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

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

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

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