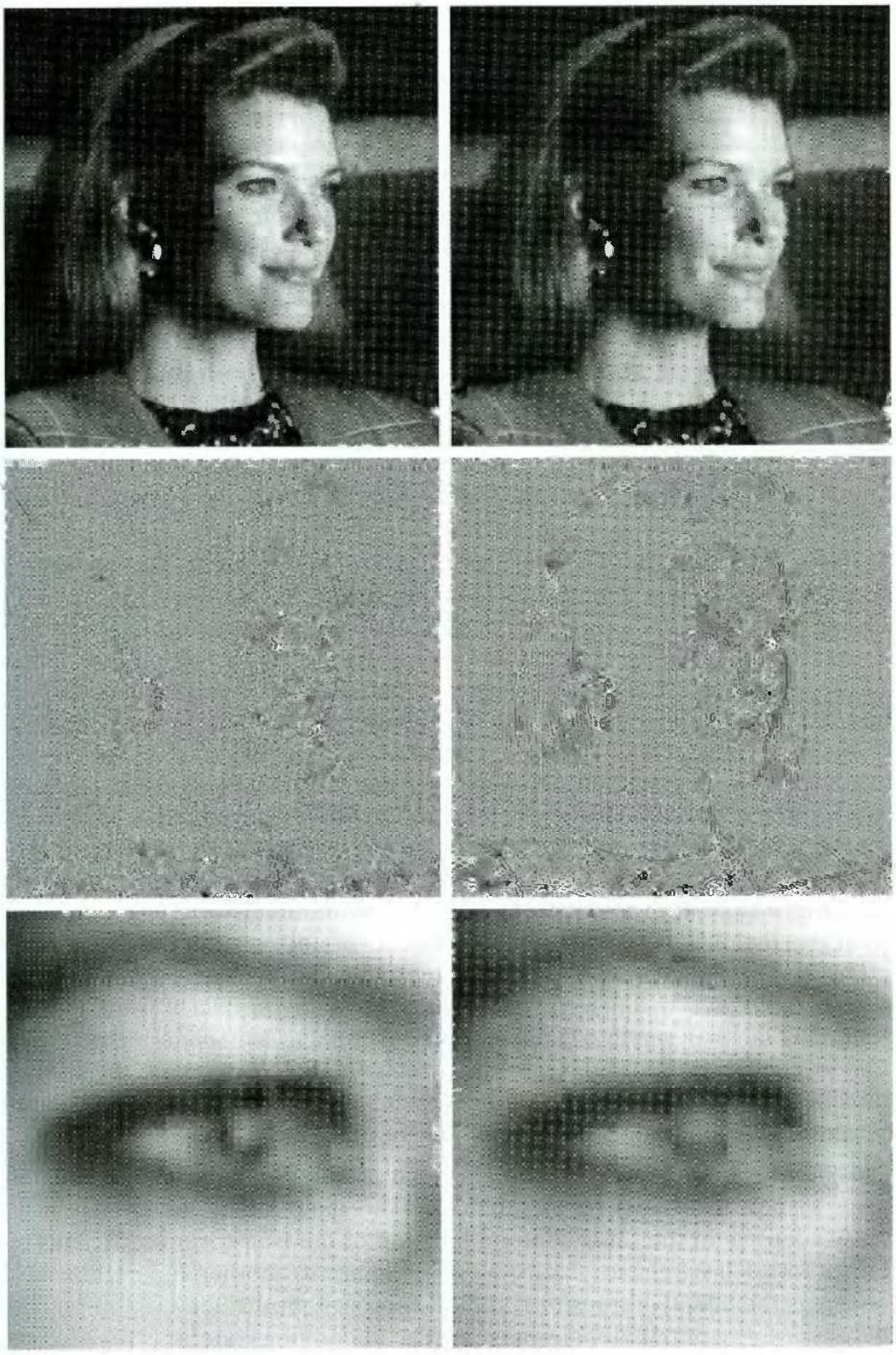


a b
c d
e f

FIGURE 8.35 Approximations of Fig. 8.23 using 12.5% of the 8×8 DCT coefficients: (a), (c), and (e) threshold coding results; (b), (d), and (f) zonal coding results.

a b
c d

FIGURE 8.36 A typical (a) zonal mask, (b) zonal bit allocation, (c) threshold mask, and (d) thresholded coefficient ordering sequence. Shading highlights the coefficients that are retained.

1	1	1	1	1	0	0	0
1	1	1	1	0	0	0	0
1	1	1	0	0	0	0	0
1	1	0	0	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

8	7	6	4	3	2	1	0
7	6	5	4	3	2	1	0
6	5	4	3	3	1	1	0
4	4	3	3	2	1	0	0
3	3	3	2	1	1	0	0
2	2	1	1	1	0	0	0
1	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0

1	1	0	1	1	0	0	0
1	1	1	1	0	0	0	0
1	1	0	0	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

modeled by a Rayleigh density function, whereas the remaining coefficients are modeled by a Laplacian or Gaussian density.[†] The number of quantization levels (and thus the number of bits) allotted to each quantizer is made proportional to $\log_2 \sigma^2_{T(u,v)}$. This allocation is consistent with rate distortion theory, which indicates that a Gaussian random variable of variance σ^2 cannot be represented by less than $\frac{1}{2}\log_2(\sigma^2/D)$ bits and be reproduced with mean-square error less than D (see Problem 8.11). The intuitive conclusion is that the information content of a Gaussian random variable is proportional to $\log_2(\sigma^2/D)$. Thus the retained coefficients in Eq. (8.5-38)—which (in the context of the current discussion) are selected on the basis of maximum variance—should be assigned bits in proportion to the logarithm of the coefficient variances.

Threshold coding implementation Zonal coding usually is implemented by using a single fixed mask for all subimages. Threshold coding, however, is inherently adaptive in the sense that the location of the transform coefficients retained for each subimage vary from one subimage to another. In fact, threshold coding is the adaptive transform coding approach most often used in practice because of its computational simplicity. The underlying concept is that, for any subimage, the transform coefficients of largest magnitude make the most significant contribution to reconstructed subimage quality, as demonstrated in the

[†] As each coefficient is a linear combination of the pixels in its subimage [see Eq. (8.5-24)], the central limit theorem suggests that, as subimage size increases, the coefficients tend to become Gaussian. This result does not apply to the dc coefficient, however, because nonnegative images always have positive dc coefficients.

last example. Because the locations of the maximum coefficients vary from one subimage to another, the elements of $\gamma(u, v)T(u, v)$ normally are reordered (in a predefined manner) to form a 1-D, run-length coded sequence. Figure 8.36(c) shows a typical *threshold mask* for one subimage of a hypothetical image. This mask provides a convenient way to visualize the threshold coding process for the corresponding subimage, as well as to mathematically describe the process using Eq. (8.5-38). When the mask is applied [via Eq. (8.5-38)] to the subimage for which it was derived, and the resulting $n \times n$ array is reordered to form an n^2-element coefficient sequence in accordance with the zigzag ordering pattern of Fig. 8.34(d), the reordered 1-D sequence contains several long runs of 0's [the zigzag pattern becomes evident by starting at 0 in Fig. 8.36(d) and following the numbers in sequence]. These runs normally are run-length coded. The nonzero or retained coefficients, corresponding to the mask locations that contain a 1, are represented using one of the variable-length codes of Section 8.4.

There are three basic ways to threshold a transformed subimage or, stated differently, to create a subimage threshold masking function of the form given in Eq. (8.5-37): (1) A single global threshold can be applied to all subimages; (2) a different threshold can be used for each subimage; or (3) the threshold can be varied as a function of the location of each coefficient within the subimage. In the first approach, the level of compression differs from image to image, depending on the number of coefficients that exceed the global threshold. In the second, called *N-largest coding*, the same number of coefficients is discarded for each subimage. As a result, the code rate is constant and known in advance. The third technique, like the first, results in a variable code rate, but offers the advantage that thresholding *and* quantization can be combined by replacing $\gamma(u, v)T(u, v)$ in Eq. (8.5-38) with

$$\hat{T}(u, v) = \text{round}\left[\frac{T(u, v)}{Z(u, v)}\right] \tag{8.5-40}$$

where $\hat{T}(u, v)$ is a thresholded and quantized approximation of $T(u, v)$, and $Z(u, v)$ is an element of the transform normalization array

$$\mathbf{Z} = \begin{bmatrix} Z(0,0) & Z(0,1) & \cdots & Z(0, n-1) \\ Z(1,0) & & & \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & & \cdot \\ Z(n-1,0) & Z(n-1,1) & \cdots & Z(n-1, n-1) \end{bmatrix}. \tag{8.5-41}$$

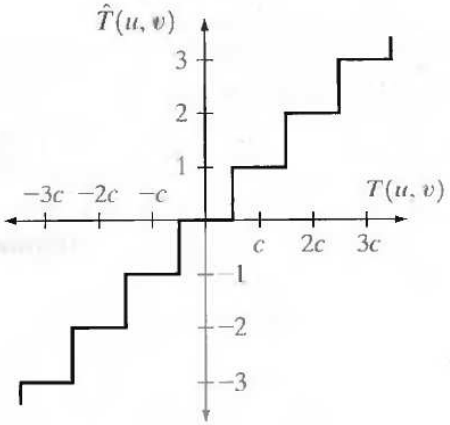
Before a normalized (thresholded and quantized) subimage transform, $\hat{T}(u, v)$, can be inverse transformed to obtain an approximation of subimage $f(x, y)$, it must be multiplied by $Z(u, v)$. The resulting denormalized array, denoted $\dot{T}(u, v)$, is an approximation of $\hat{T}(u, v)$:

$$\dot{T}(u, v) = \hat{T}(u, v)Z(u, v). \tag{8.5-42}$$

The inverse transform of $\dot{T}(u, v)$ yields the decompressed subimage approximation.

a b

FIGURE 8.37 (a) A threshold coding quantization curve [see Eq. (8.5-40)]. (b) A typical normalization matrix.



16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Figure 8.37(a) depicts Eq. (8.5-40) graphically for the case in which $Z(u, v)$ is assigned a particular value c. Note that $\hat{T}(u, v)$ assumes integer value k if and only if

$$kc - \frac{c}{2} \leq T(u, v) < kc + \frac{c}{2}.$$

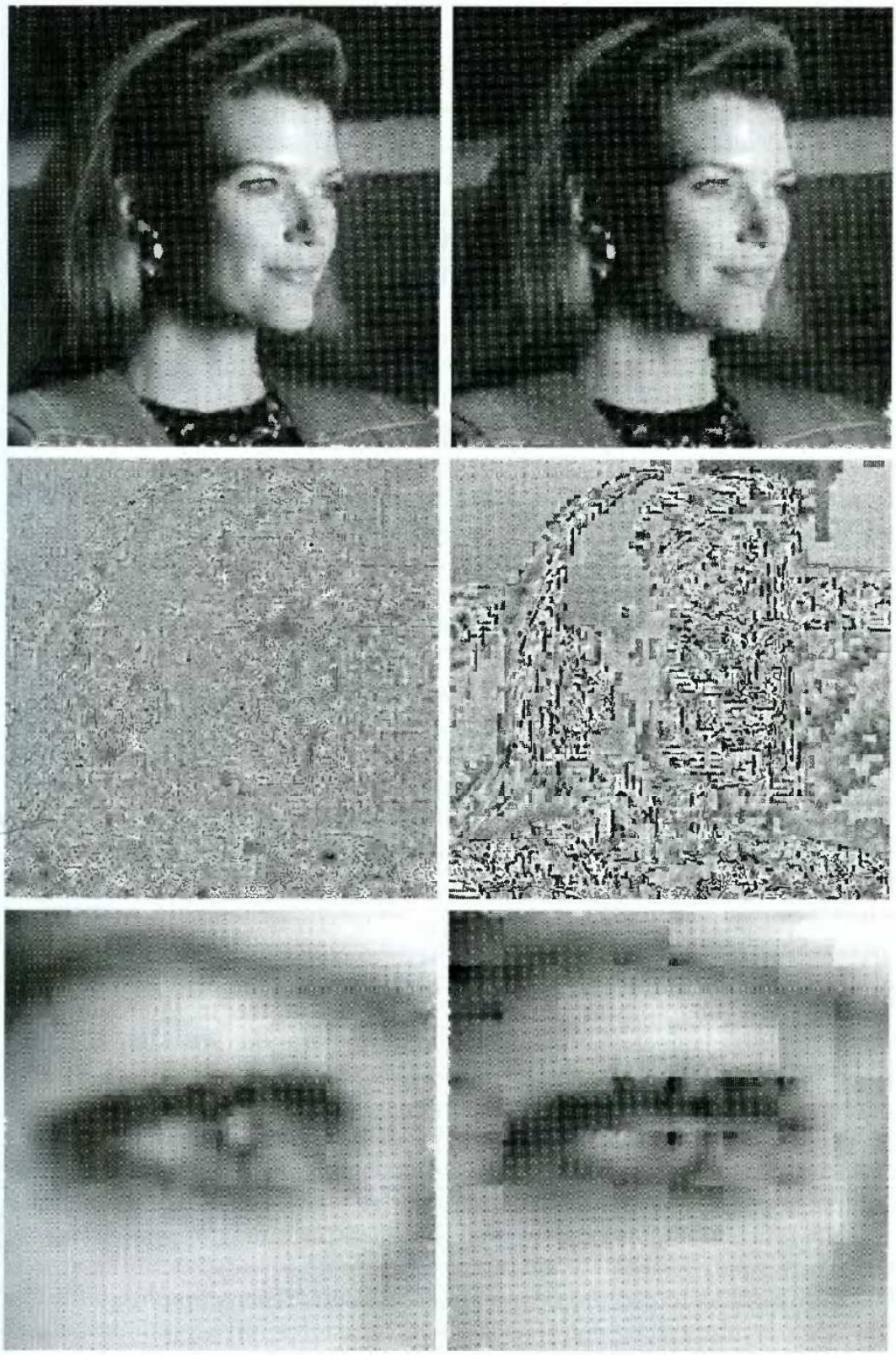
If $Z(u, v) > 2T(u, v)$, then $\hat{T}(u, v) = 0$ and the transform coefficient is completely truncated or discarded. When $\hat{T}(u, v)$ is represented with a variable-length code that increases in length as the magnitude of k increases, the number of bits used to represent $T(u, v)$ is controlled by the value of c. Thus the elements of $\mathbf{Z}$ can be scaled to achieve a variety of compression levels. Figure 8.37(b) shows a typical normalization array. This array, which has been used extensively in the JPEG[†] standardization efforts (see Section 8.6.2), weighs each coefficient of a transformed subimage according to heuristically determined perceptual or psychovisual importance.

EXAMPLE 8.22: Illustration of threshold coding.

■ Figures 8.38(a) and (b) show two threshold-coded approximations of the monochrome image in Fig. 8.23. Both images were generated using an 8×8 DCT and the normalization array of Fig. 8.37(b). The first result, which provides a compression ratio of about 34 to 1, was obtained by direct application of that normalization array. The second, which compresses the original image by a ratio of 67 to 1, was generated after multiplying (scaling) the normalization array by 4. By comparison, the *average* compression ratio obtained by using all the error-free methods discussed in Section 8.4 was only 2.62 to 1.

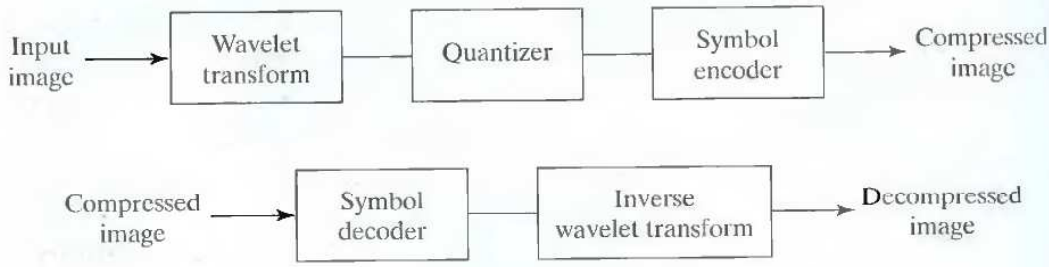
The differences between the original image of Fig. 8.23 and the reconstructed images of Figs. 8.38(a) and (b) are shown in Figs. 8.38(c) and (d), respectively. The corresponding rms errors [see Eq. (8.1-8)] are 3.42 and 6.33 gray levels. The precise nature of the errors is more visible in the zoomed images of Figures 8.38(e) and (f). These images show a magnified section of Figs. 8.38(a) and (b), respectively. They allow a better assessment of the subtle differences between the reconstructed images. ■

[†] JPEG is an abbreviation for the Joint Photographic Experts Group.



a b
c d
e f

FIGURE 8.38 Left column: Approximations of Fig. 8.23 using the DCT and normalization array of Fig. 8.37(b). Right column: Similar results for 4**Z**.



a
b

FIGURE 8.39 A wavelet coding system: (a) encoder; (b) decoder.

8.5.3 Wavelet Coding

Like the transform coding techniques of the previous section, wavelet coding is based on the idea that the coefficients of a transform that decorrelates the pixels of an image can be coded more efficiently than the original pixels themselves. If the transform's basis functions—in this case wavelets—pack most of the important visual information into a small number of coefficients, the remaining coefficients can be quantized coarsely or truncated to zero with little image distortion.

Figure 8.39 shows a typical wavelet coding system. To encode a $2^J \times 2^J$ image, an analyzing wavelet, ψ, and minimum decomposition level, $J - P$, are selected and used to compute the image's discrete wavelet transform. If the wavelet has a complimentary scaling function φ, the fast wavelet transform (see Sections 7.4 and 7.5) can be used. In either case, the computed transform converts a large portion of the original image to horizontal, vertical, and diagonal decomposition coefficients with zero mean and Laplacian-like distributions. Recall the image of Fig. 7.1 and the dramatically simpler statistics of its wavelet transform in Fig. 7.8(a). Since many of the computed coefficients carry little visual information, they can be quantized and coded to minimize intercoefficient and coding redundancy. Moreover, the quantization can be adapted to exploit any positional correlation across the P decomposition levels. One or more of the lossless coding methods of Section 8.4, including run-length, Huffman, arithmetic, and bit-plane coding, can be incorporated into the final symbol coding step. Decoding is accomplished by inverting the encoding operations—with the exception of quantization, which cannot be reversed exactly.

The principal difference between the wavelet-based system of Fig. 8.39 and the transform coding system of Fig. 8.28 is the omission of the transform coder's subimage processing stages. Because wavelet transforms are both computationally efficient and inherently local (i.e., their basis functions are limited in duration), subdivision of the original image is unnecessary. As will be seen in the following example, the removal of the subdivision step eliminates the blocking artifact that characterizes DCT-based approximations at high compression ratios.

EXAMPLE 8.23: Comparing wavelet and DCT-based coding.

■ Figure 8.40 shows two wavelet-based approximations of the monochrome image in Figure 8.23. Figure 8.40(a) was reconstructed from an encoding that compressed the original image by 34 : 1; Fig. 8.40(b) was generated from a 67 : 1 encoding. Since these ratios are identical to the compression levels of

Example 8.22, Figs. 8.40(a) through (f) can be compared—both qualitatively and quantitatively—to the transform-based results in Figs. 8.38(a) through (f).

A visual comparison of Figs. 8.40(c) and (d) with 8.38(c) and (d), respectively, reveals a noticeable decrease of error in the wavelet coding results. In fact, the rms error of the wavelet-based image in Fig. 8.40(a) is 2.29 gray levels, as opposed to 3.42 gray levels for the corresponding transform-based result in Fig. 8.38(a). In a similar manner, the rms errors of the approximations in Figs. 8.38(b) and 8.40(b) are 6.33 and 2.96, respectively. The computed errors favor the wavelet-based results at both compression levels.

Besides decreasing the reconstruction error for a given level of compression, wavelet coding—see Figs. 8.40(e) and (f)—dramatically increases (in a subjective sense) image quality. This is particularly evident in Fig. 8.40(f). Note that the blocking artifact that dominated the corresponding transform-based result in Fig. 8.38(f) is no longer present. ■

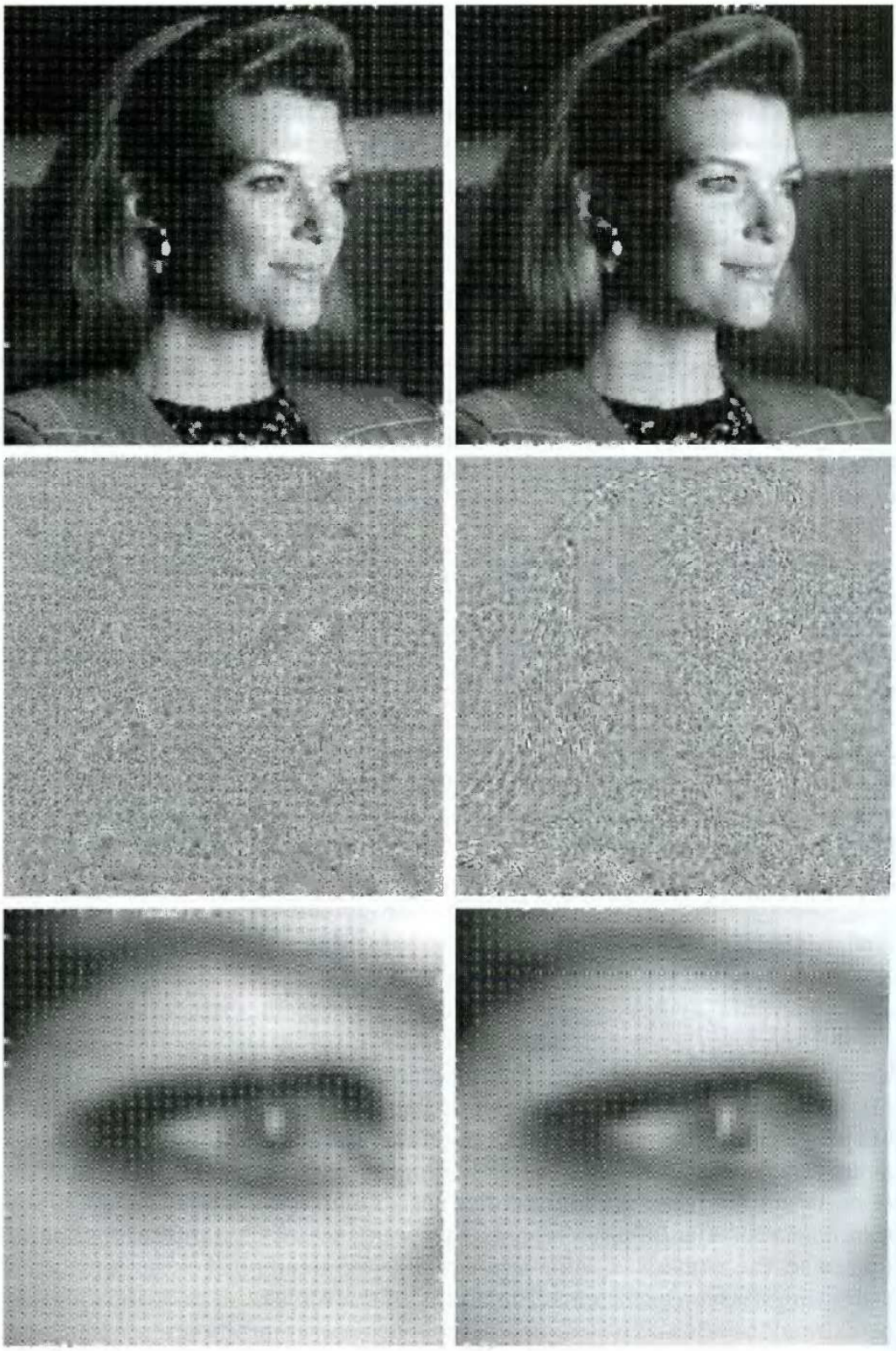
When the level of compression exceeds 67:1, the largest ratio examined in the previous two examples, there is an increased loss of texture in the woman's clothing and blurring of her eyes. Both effects are visible in Figs. 8.41(a) and (b), which were reconstructed from 108:1 and 167:1 wavelet-based encodings of the original image in Fig. 8.23. The increased blurring is particularly evident in Figs. 8.41(e) and (f). The rms errors for Figs. 8.41(a) and (b) are 3.72 and 4.73 gray levels, respectively. A subjective evaluation of either image reveals its obvious superiority to the 67:1 transform-based result in Fig. 8.38(b). Its rms error was 6.33 gray levels. Thus, at more than twice the level of compression, the most highly compressed wavelet-based reconstruction has only 75% of the error of the less compressed transform-based result—and superior perceived quality as well.

To conclude our discussion of wavelet-based compression, we end with a brief overview of the major factors influencing coding complexity, performance, and reconstruction error.

Wavelet selection

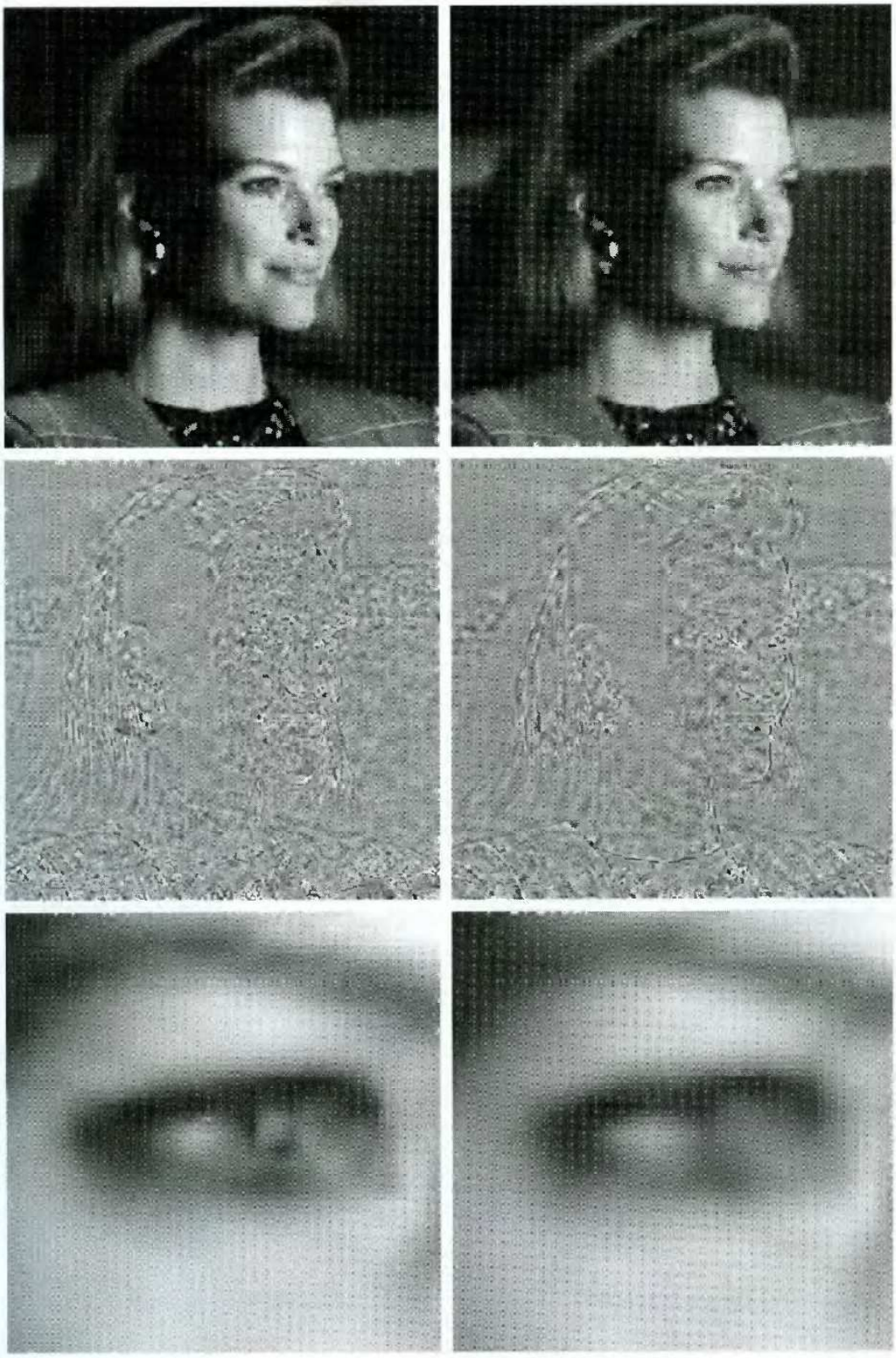
The wavelets chosen as the basis of the forward and inverse transforms in Fig. 8.39 affect all aspects of wavelet coding system design and performance. They impact directly the computational complexity of the transforms and, less directly, the system's ability to compress and reconstruct images of acceptable error. When the transforming wavelet has a companion scaling function, the transformation can be implemented as a sequence of digital filtering operations, with the number of filter taps equal to the number of nonzero wavelet and scaling vector coefficients. The ability of the wavelet to pack information into a small number of transform coefficients determines its compression and reconstruction performance.

The most widely used expansion functions for wavelet-based compression are the Daubechies wavelets and biorthogonal wavelets. The latter allow useful analysis properties, like number of zero moments (see Section 7.5), to be incorporated into the decomposition filters, while important synthesis properties, like smoothness of reconstruction, are built into the reconstruction filters.



a b
c d
e f

FIGURE 8.40 (a), (c), and (e) Wavelet coding results comparable to the transform-based results in Figs. 8.38(a), (c), and (e); (b), (d), and (f) similar results for Figs. 8.38(b), (d), and (f).



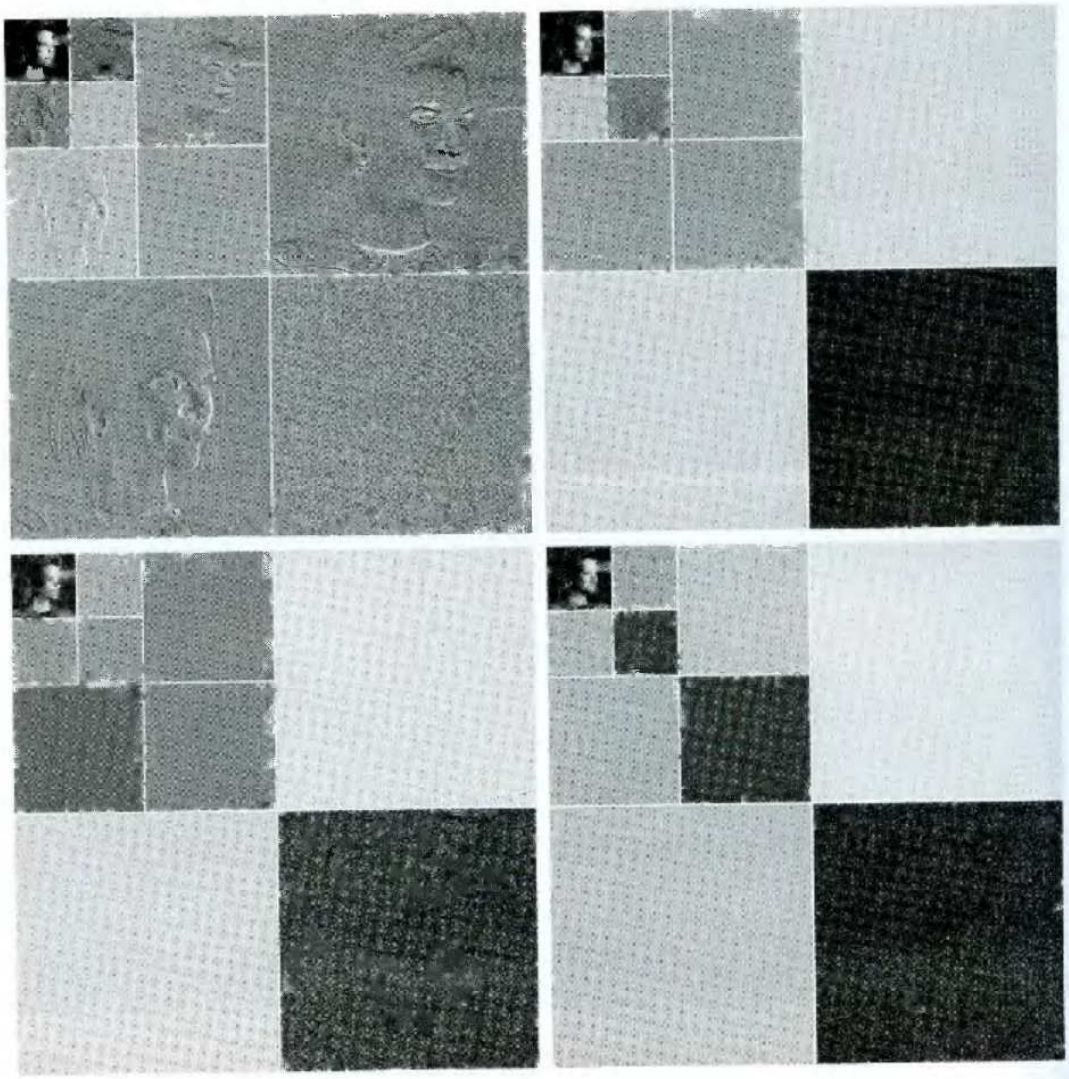
a b
c d
e f

FIGURE 8.41 (a), (c), and (e) Wavelet coding results with a compression ratio of 108 to 1; (b), (d), and (f) similar results for a compression of 167 to 1.

EXAMPLE 8.24: Wavelet bases in wavelet coding.

■ Figure 8.42 contains four discrete wavelet transforms of the woman in Fig. 8.23. Haar wavelets, the simplest and only discontinuous wavelets considered here, were used as the expansion or basis functions in Fig. 8.42(a). Daubechies wavelets, among the most popular imaging wavelets, were used in Fig. 8.42(b), and symlets, which are an extension of the Daubechies wavelets with increased symmetry, were used in Fig. 8.42(c). The Cohen-Daubechies-Feauveau wavelets that were employed in Fig. 8.42(d) are included to illustrate the capabilities of biorthogonal wavelets. As in previous results of this type, all detail coefficients have been scaled to make the underlying structure more visible—with gray-level 128 corresponding to coefficient value 0.

As can be seen in Table 8.12, the number of operations involved in the computation of the transforms in Fig. 8.42 increases from 4 to 28 multiplications



a b
c d

FIGURE 8.42 Wavelet transforms of Fig. 8.23 with respect to (a) Haar wavelets, (b) Daubechies wavelets, (c) symlets, and (d) Cohen-Daubechies-Feauveau biorthogonal wavelets.

Wavelet	Filter Taps (Scaling + Wavelet)	Zeroed Coefficients
Haar (see Ex. 7.10)	2 + 2	46%
Daubechies (see Fig. 7.6)	8 + 8	51%
Symlet (see Fig. 7.24)	8 + 8	51%
Biorthogonal (see Fig. 7.37)	17 + 11	55%

TABLE 8.12 Wavelet transform filter taps and zeroed coefficients when truncating the transforms in Fig. 8.42 below 1.5.

and additions per coefficient (for each decomposition level) as you move from Fig. 8.42(a) to (d). All four transforms were computed using a fast wavelet transform (i.e., filter bank) formulation. Note that as the computational complexity (i.e., the number of filter taps) increases, the information packing ability does as well. When Haar wavelets are employed and the detail coefficients below 1.5 are truncated to zero, 46% of the total transform is zeroed. With the more complex biorthogonal wavelets, the number of zeroed coefficients rises to 55%, increasing the potential compression by almost 10%. ■

Decomposition level selection

Another factor affecting wavelet coding computational complexity and reconstruction error is the number of transform decomposition levels. Since a P-scale fast wavelet transform involves P filter bank iterations, the number of operations in the computation of the forward and inverse transforms increases with the number of decomposition levels. Moreover, quantizing the increasingly lower-scale coefficients that result with more decomposition levels impacts increasingly larger areas of the reconstructed image. In many applications, like searching image databases or transmitting images for progressive reconstruction, the resolution of the stored or transmitted images and the scale of the lowest useful approximations normally determine the number of transform levels.

EXAMPLE 8.25: Decomposition levels in wavelet coding.

■ Table 8.13 illustrates the effect of decomposition level selection on the coding of Fig. 8.23 with a fixed global threshold of 25. As in previous wavelet coding examples, only detail coefficients are truncated. The table lists both the percentage of zeroed coefficients and the resulting rms reconstruction errors. Note that the initial decompositions are responsible for the majority of the data compression. There is little change in the number of truncated coefficients above three decomposition levels. ■

Scales and Filter Bank Iterations	Approximation Coefficient Image	Truncated Coefficients (%)	Reconstruction Error (rms)
1	256 × 256	75%	1.93
2	128 × 128	93%	2.69
3	64 × 64	97%	3.12
4	32 × 32	98%	3.25
5	16 × 16	98%	3.27

TABLE 8.13 Decomposition level impact on wavelet coding the 512 × 512 image of Fig. 8.23.

TABLE 8.14 CCITT terminating codes.

Run Length	White Code Word	Black Code Word	Run Length	White Code Word	Black Code Word
0	00110101	0000110111	32	00011011	000001101010
1	000111	010	33	00010010	000001101011
2	0111	11	34	00010011	000011010010
3	1000	10	35	00010100	000011010011
4	1011	011	36	00010101	000011010100
5	1100	0011	37	00010110	000011010101
6	1110	0010	38	00010111	000011010110
7	1111	00011	39	00101000	000011010111
8	10011	000101	40	00101001	000001101100
9	10100	000100	41	00101010	000001101101
10	00111	0000100	42	00101011	000011011010
11	01000	0000101	43	00101100	000011011011
12	001000	0000111	44	00101101	000001010100
13	000011	00000100	45	00000100	000001010101
14	110100	00000111	46	00000101	000001010110
15	110101	000011000	47	00001010	000001010111
16	101010	0000010111	48	00001011	000001100100
17	101011	0000011000	49	01010010	000001100101
18	0100111	0000001000	50	01010011	000001010010
19	0001100	00001100111	51	01010100	000001010011
20	0001000	00001101000	52	01010101	000000100100
21	0010111	00001101100	53	00100100	000000110111
22	0000011	00000110111	54	00100101	000000111000
23	0000100	00000101000	55	01011000	000000100111
24	0101000	00000010111	56	01011001	000000101000
25	0101011	00000011000	57	01011010	000001011000
26	0010011	000011001010	58	01011011	000001011001
27	0100100	000011001011	59	01001010	000000101011
28	0011000	000011001100	60	01001011	000000101100
29	00000010	000011001101	61	00110010	000001011010
30	00000011	000001101000	62	00110011	000001100110
31	00011010	000001101001	63	00110100	000001100111

a unique end-of-line (EOL) code word 000000000001 is used to terminate each line, as well as to signal the first line of each new image. The end of a sequence of images is indicated by six consecutive EOLs.

Two-dimensional compression

The 2-D compression approach adopted for both the CCITT Group 3 and 4 standards is a line-by-line method in which the position of each black-to-white or white-to-black run transition is coded with respect to the position of a *reference element* a_0 that is situated on the current *coding line*. The previously coded line is called the *reference line*; the reference line for the first line of each new image is an imaginary white line.

TABLE 8.15 CCITT makeup codes.

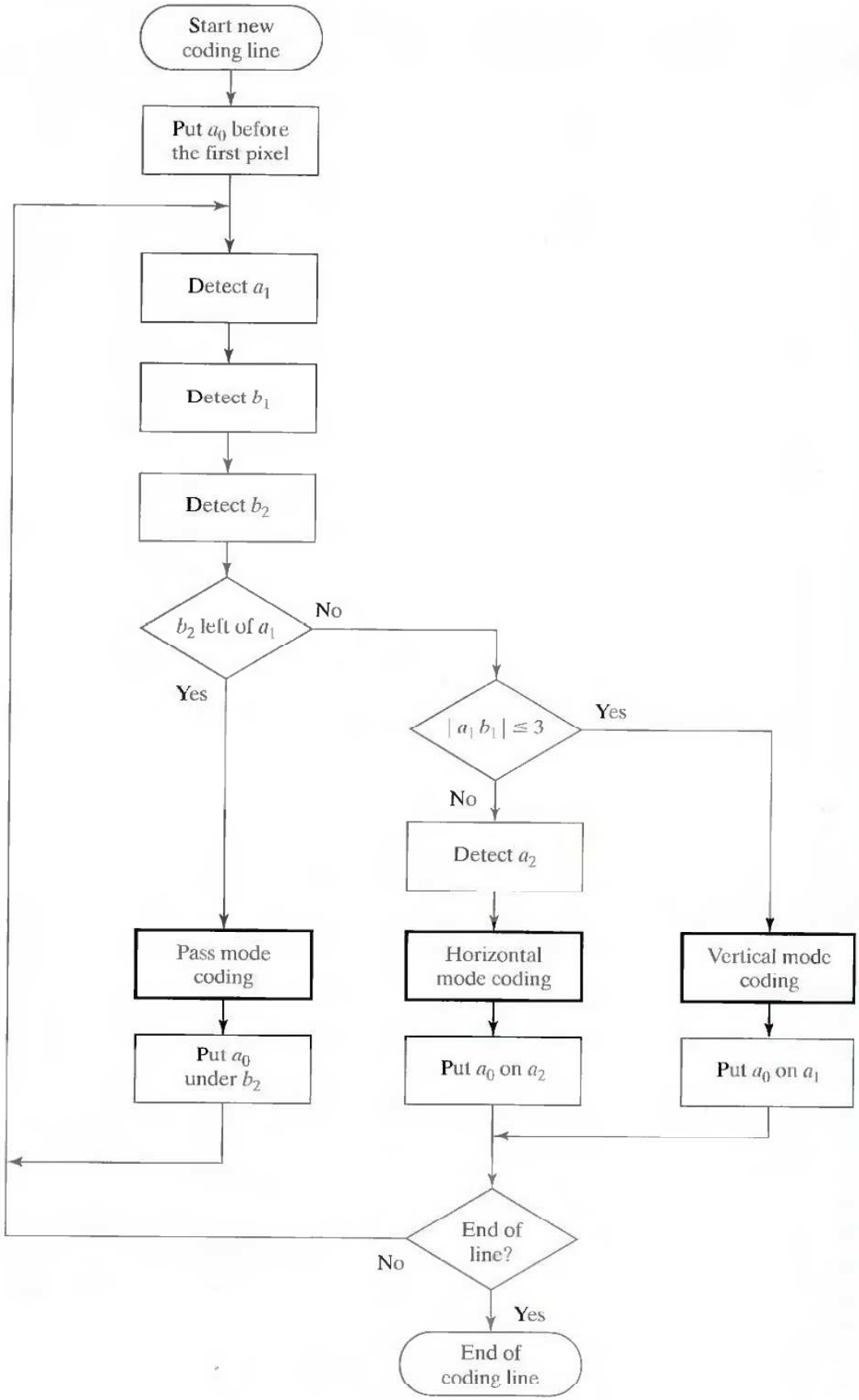
Run Length	White Code Word	Black Code Word	Run Length	White Code Word	Black Code Word
64	11011	0000001111	960	011010100	0000001110011
128	10010	000011001000	1024	011010101	0000001110100
192	010111	000011001001	1088	011010110	0000001110101
256	0110111	000001011011	1152	011010111	0000001110110
320	00110110	000000110011	1216	011011000	0000001110111
384	00110111	000000110100	1280	011011001	0000001010010
448	01100100	000000110101	1344	011011010	0000001010011
512	01100101	0000001101100	1408	011011011	0000001010100
576	01101000	0000001101101	1472	010011000	0000001010101
640	01100111	0000001001010	1536	010011001	0000001011010
704	011001100	0000001001011	1600	010011010	0000001011011
768	011001101	0000001001100	1664	011000	0000001100100
832	011010010	0000001001101	1728	010011011	0000001100101
896	011010011	0000001110010			

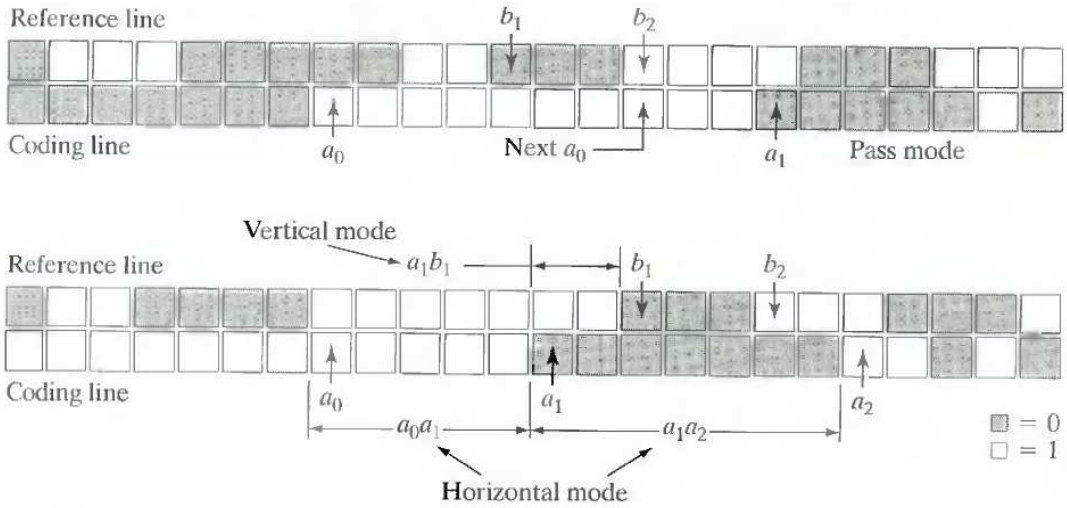
	Code Word		Code Word
1792	00000001000	2240	000000010110
1856	00000001100	2304	000000010111
1920	00000001101	2368	000000011100
1984	000000010010	2432	000000011101
2048	000000010011	2496	000000011110
2112	000000010100	2560	000000011111
2176	000000010101		

Figure 8.44 shows the basic coding process for a single scan line. Note that the initial steps of the procedure are directed at locating several key transition or *changing elements*: a_0, a_1, a_2, b_1, and b_2. A changing element is defined as a pixel whose value is different from that of the previous pixel on the same line. The most important changing element is a_0 (the reference element), which is either set to the location of an imaginary white changing element to the left of the first pixel of each new coding line or determined from the previous coding mode. After a_0 is located, a_1 is identified as the location of the next changing element to the right of a_0 on the current coding line, a_2 as the next changing element to the right of a_1 on the coding line, b_1 as the changing element of the opposite value (of a_0) and to the right of a_0 on the reference (or previous) line, and b_2 as the next changing element to the right of b_1 on the reference line. If any of these changing elements are not detected, they are set to the location of an imaginary pixel to the right of the last pixel on the appropriate line. Figure 8.45 provides two illustrations of the general relationships between the various changing elements.

After identification of the current reference element and associated changing elements, two simple tests are performed to select one of three possible coding modes: *pass mode, vertical mode*, or *horizontal mode*. The inital test, which

FIGURE 8.44 CCITT 2-D coding procedure. The notation $|a_1 b_1|$ denotes the absolute value of the distance between changing elements a_1 and b_1.





a
b

FIGURE 8.45 CCITT (a) pass mode and (b) horizontal and vertical mode coding parameters.

corresponds to the first branch point in the flow chart shown in Fig. 8.44, compares the location of b_2 to that of a_1. The second test, which corresponds to the second branch point in Fig. 8.44, computes the distance (in pixels) between the locations of a_1 and b_1 and compares it against 3. Depending on the outcome of these tests, one of the three outlined coding blocks of Fig. 8.44 is entered and the appropriate coding procedure is executed. A new reference element is then established, as per the flow chart, in preparation for the next coding iteration.

Table 8.16 defines the specific codes utilized for each of the three possible coding modes. In pass mode, which specifically excludes the case in which b_2 is directly above a_1, only the pass mode code word 0001 is needed. As Fig. 8.45(a) shows, this mode identifies white or black reference line runs that do not overlap the current white or black coding line runs. In horizontal coding mode, the distances from a_0 to a_1 and a_1 to a_2 must be coded in accordance with the termination and makeup codes of Tables 8.14 and 8.15 and then appended to the horizontal mode code word 001. This is indicated in Table 8.16 by the notation $001 + M(a_0a_1) + M(a_1a_2)$, where a_0a_1 and a_1a_2 denote the distances from a_0 to a_1 and a_1 to a_2, respectively. Finally, in vertical coding mode, one of six special variable-length codes is assigned to the distance between a_1 and b_1. Figure 8.45(b) illustrates the parameters involved in both horizontal and vertical mode coding. The *extension mode* code word at the bottom of Table 8.16 is used to enter an optional facsimile coding mode. For example, the 0000001111 code is used to initiate an uncompressed mode of transmission.

EXAMPLE 8.27: CCITT vertical mode coding example.

■ Although Fig. 8.45(b) is annotated with the parameters for both horizontal and vertical mode coding, the depicted condition in reality is a case for vertical mode coding. That is, as b_2 is to the right of a_1, the first (or pass mode) test in Fig. 8.44 fails. The second test, which determines whether the vertical or horizontal coding mode is entered, indicates that vertical mode coding should be used, because the distance from a_1 to b_1 is less than 3. In accordance with Table 8.16, the appropriate code word is 000010, implying that a_1 is two pixels left of b_1. In preparation for the next coding iteration, a_0 is moved to the location of a_1. ■

TABLE 8.16
CCITT two-dimensional code table.

Mode	Code Word
Pass	0001
Horizontal	$001 + M(a_0a_1) + M(a_1a_2)$
Vertical	
a_1 below b_1	1
a_1 one to the right of b_1	011
a_1 two to the right of b_1	000011
a_1 three to the right of b_1	0000011
a_1 one to the left of b_1	010
a_1 two to the left of b_1	000010
a_1 three to the left of b_1	0000010
Extension	0000001xxx

8.6.2 Continuous Tone Still Image Compression Standards

The CCITT and ISO have defined several continuous tone (as opposed to binary) image compression standards. These standards, which are in various phases of the adoption process, address both monochrome and color image compression. In contrast to the binary compression standards described in Section 8.6.1, continuous tone standards are based principally on the lossy transform coding techniques of Sections 8.5.2 and 8.5.3. To develop the standards, CCITT and ISO committees solicited algorithm recommendations from a large number of companies, universities, and research laboratories. The best of those submitted were selected on the basis of image quality and compression performance. The resulting standards, which include the original DCT-based JPEG standard, the recently proposed wavelet-based JPEG 2000 standard, and the JPEG-LS standard, a lossless to near lossless adaptive prediction scheme that includes a mechanism for flat region detection and run-length coding (ISO/IEC [1999]), represent the state of the art in continuous tone image compression.

JPEG

One of the most popular and comprehensive continuous tone, still frame compression standards is the JPEG standard. It defines three different coding systems: (1) a lossy *baseline coding system*, which is based on the DCT and is adequate for most compression applications; (2) an *extended coding system* for greater compression, higher precision, or progressive reconstruction applications; and (3) a lossless *independent coding system* for reversible compression. To be JPEG compatible, a product or system must include support for the baseline system. No particular file format, spatial resolution, or color space model is specified.

In the baseline system, often called the *sequential baseline system*, the input and output data precision is limited to 8 bits, whereas the quantized DCT values are restricted to 11 bits. The compression itself is performed in three sequential steps: DCT computation, quantization, and variable-length code assignment. The image is first subdivided into pixel blocks of size 8×8, which are processed left to right, top to bottom. As each 8×8 block or subimage is encountered, its 64 pixels are level shifted by subtracting the quantity 2^{n-1}, where 2^n is the maximum number of

gray levels. The 2-D discrete cosine transform of the block is then computed, quantized in accordance with Eq. (8.5-40), and reordered, using the zigzag pattern of Fig. 8.36(d), to form a 1-D sequence of quantized coefficients.

Since the one-dimensionally reordered array generated under the zigzag pattern of Fig. 8.36(d) is qualitatively arranged according to increasing spatial frequency, the JPEG coding procedure is designed to take advantage of the long runs of zeros that normally result from the reordering. In particular, the nonzero AC[†] coefficients are coded using a variable-length code that defines the coefficient's value and number of preceding zeros. The DC coefficient is difference coded relative to the DC coefficient of the previous subimage. Tables 8.17, 8.18, and 8.19 provide the default JPEG Huffman codes for luminance imagery. The JPEG recommended luminance quantization array is given in Fig. 8.37(b) and can be scaled to provide a variety of compression levels. Although default coding tables and proven quantization arrays are provided for both luminance and chrominance processing, the user is free to construct custom tables and/or arrays, which may in fact be adapted to the characteristics of the image being compressed.

EXAMPLE 8.28: JPEG baseline coding and decoding.

■ Consider compression and reconstruction of the following 8×8 subimage with the JPEG baseline standard:

52	55	61	66	70	61	64	73
63	59	66	90	109	85	69	72
62	59	68	113	144	104	66	73
63	58	71	122	154	106	70	69
67	61	68	104	126	88	68	70
79	65	60	70	77	68	58	75
85	71	64	59	55	61	65	83
87	79	69	68	65	76	78	94

The original image consists of 256 or 2^8 possible gray levels, so the coding process begins by level shifting the pixels of the original subimage by -2^7 or -128 gray levels. The resulting shifted array is

−76	−73	−67	−62	−58	−67	−64	−55
−65	−69	−62	−38	−19	−43	−59	−56
−66	−69	−60	−15	16	−24	−62	−55
−65	−70	−57	−6	26	−22	−58	−59
−61	−67	−60	−24	−2	−40	−60	−58
−49	−63	−68	−58	−51	−65	−70	−53
−43	−57	−64	−69	−73	−67	−63	−45
−41	−49	−59	−60	63	−52	−50	−34

[†] In the standard, the term AC denotes all transform coefficients with the exception of the zeroth or DC coefficient.

TABLE 8.17
JPEG coefficient coding categories.

Range	DC Difference Category	AC Category
0	0	N/A
−1, 1	1	1
−3, −2, 2, 3	2	2
−7, ..., −4, 4, ..., 7	3	3
−15, ..., −8, 8, ..., 15	4	4
−31, ..., −16, 16, ..., 31	5	5
−63, ..., −32, 32, ..., 63	6	6
−127, ..., −64, 64, ..., 127	7	7
−255, ..., −128, 128, ..., 255	8	8
−511, ..., −256, 256, ..., 511	9	9
−1023, ..., −512, 512, ..., 1023	A	A
−2047, ..., −1024, 1024, ..., 2047	B	B
−4095, ..., −2048, 2048, ..., 4095	C	C
−8191, ..., −4096, 4096, ..., 8191	D	D
−16383, ..., −8192, 8192, ..., 16383	E	E
−32767, ..., −16384, 16384, ..., 32767	F	N/A

TABLE 8.18
JPEG default DC code (luminance).

Category	Base Code	Length	Category	Base Code	Length
0	010	3	6	1110	10
1	011	4	7	11110	12
2	100	5	8	111110	14
3	00	5	9	1111110	16
4	101	7	A	11111110	18
5	110	8	B	111111110	20

which, when transformed in accordance with the forward DCT of Eqs. (8.5-24) and (8.5-32) for $N = 8$, becomes

$$
\begin{matrix}
-415 & -29 & -62 & 25 & 55 & -20 & -1 & 3 \\
7 & -21 & -62 & 9 & 11 & -7 & -6 & 6 \\
-46 & 8 & 77 & -25 & -30 & 10 & 7 & -5 \\
-50 & 13 & 35 & -15 & -9 & 6 & 0 & 3 \\
11 & -8 & -13 & -2 & -1 & 1 & -4 & 1 \\
-10 & 1 & 3 & -3 & -1 & 0 & 2 & -1 \\
-4 & -1 & 2 & -1 & 2 & -3 & 1 & -2 \\
-1 & -1 & -1 & -2 & -1 & -1 & 0 & -1
\end{matrix}
$$

TABLE 8.19
JPEG default AC code (luminance) *(continues on next page)*.

Run/ Category	Base Code	Length	Run/ Category	Base Code	Length
0/0	**1010 (= EOB)**	**4**			
0/1	00	3	8/1	11111010	9
0/2	01	4	8/2	111111111000000	17
0/3	100	6	8/3	1111111110110111	19
0/4	1011	8	8/4	1111111110111000	20
0/5	11010	10	8/5	1111111110111001	21
0/6	111000	12	8/6	1111111110111010	22
0/7	1111000	14	8/7	1111111110111011	23
0/8	1111110110	18	8/8	1111111110111100	24
0/9	1111111110000010	25	8/9	1111111110111101	25
0/A	1111111110000011	26	8/A	1111111110111110	26
1/1	1100	5	9/1	111111000	10
1/2	111001	8	9/2	1111111110111111	18
1/3	1111001	10	9/3	1111111111000000	19
1/4	111110110	13	9/4	1111111111000001	20
1/5	11111110110	16	9/5	1111111111000010	21
1/6	1111111110000100	22	9/6	1111111111000011	22
1/7	1111111110000101	23	9/7	1111111111000100	23
1/8	1111111110000110	24	9/8	1111111111000101	24
1/9	1111111110000111	25	9/9	1111111111000110	25
1/A	1111111110001000	26	9/A	1111111111000111	26
2/1	11011	6	A/1	111111001	10
2/2	11111000	10	A/2	1111111111001000	18
2/3	1111110111	13	A/3	1111111111001001	19
2/4	1111111110001001	20	A/4	1111111111001010	20
2/5	1111111110001010	21	A/5	1111111111001011	21
2/6	1111111110001011	22	A/6	1111111111001100	22
2/7	1111111110001100	23	A/7	1111111111001101	23
2/8	1111111110001101	24	A/8	1111111111001110	24
2/9	1111111110001110	25	A/9	1111111111001111	25
2/A	1111111110001111	26	A/A	1111111111010000	26
3/1	111010	7	B/1	111111010	10
3/2	111110111	11	B/2	1111111111010001	18
3/3	11111110111	14	B/3	1111111111010010	19
3/4	1111111110010000	20	B/4	1111111111010011	20
3/5	1111111110010001	21	B/5	1111111111010100	21
3/6	1111111110010010	22	B/6	1111111111010101	22
3/7	1111111110010011	23	B/7	1111111111010110	23
3/8	1111111110010100	24	B/8	1111111111010111	24
3/9	1111111110010101	25	B/9	1111111111011000	25
3/A	1111111110010110	26	B/A	1111111111011001	26
4/1	111011	7	C/1	1111111010	11
4/2	1111111000	12	C/2	1111111111011010	18
4/3	1111111110010111	19	C/3	1111111111011011	19
4/4	1111111110011000	20	C/4	1111111111011100	20
4/5	1111111110011001	21	C/5	1111111111011101	21
4/6	1111111110011010	22	C/6	1111111111011110	22
4/7	1111111110011011	23	C/7	1111111111011111	23
4/8	1111111110011100	24	C/8	1111111111100000	24
4/9	1111111110011101	25	C/9	1111111111100001	25
4/A	1111111110011110	26	C/A	1111111111100010	26

TABLE 8.19
JPEG default AC code (luminance) *(continued)*.

Run/ Category	Base Code	Length	Run/ Category	Base Code	Length
5/1	1111010	8	D/1	11111111010	12
5/2	1111111001	12	D/2	1111111111100011	18
5/3	1111111110011111	19	D/3	1111111111100100	19
5/4	1111111110100000	20	D/4	1111111111100101	20
5/5	1111111110100001	21	D/5	1111111111100110	21
5/6	1111111110100010	22	D/6	1111111111100111	22
5/7	1111111110100011	23	D/7	1111111111101000	23
5/8	1111111110100100	24	D/8	1111111111101001	24
5/9	1111111110100101	25	D/9	1111111111101010	25
5/A	1111111110100110	26	D/A	1111111111101011	26
6/1	1111011	8	E/1	111111110110	13
6/2	11111111000	13	E/2	1111111111101100	18
6/3	1111111110100111	19	E/3	1111111111101101	19
6/4	1111111110101000	20	E/4	1111111111101110	20
6/5	1111111110101001	21	E/5	1111111111101111	21
6/6	1111111110101010	22	E/6	1111111111110000	22
6/7	1111111110101011	23	E/7	1111111111110001	23
6/8	1111111110101100	24	E/8	1111111111110010	24
6/9	1111111110101101	25	E/9	1111111111110011	25
6/A	1111111110101110	26	E/A	1111111111110100	26
7/1	11111001	9	**F/0**	**111111110111**	**12**
7/2	11111111001	13	F/1	1111111111110101	17
7/3	1111111110101111	19	F/2	1111111111110110	18
7/4	1111111110110000	20	F/3	1111111111110111	19
7/5	1111111110110001	21	F/4	1111111111111000	20
7/6	1111111110110010	22	F/5	1111111111111001	21
7/7	1111111110110011	23	F/6	1111111111111010	22
7/8	1111111110110100	24	F/7	1111111111111011	23
7/9	1111111110110101	25	F/8	1111111111111100	24
7/A	1111111110110110	26	F/9	1111111111111101	25
			F/A	1111111111111110	26

If the JPEG recommended normalization array of Fig. 8.37(b) is used to quantize the transformed array, the scaled and truncated [that is, normalized in accordance with Eq. (8.5-40)] coefficients are

$$
\begin{matrix}
-26 & -3 & -6 & 2 & 2 & 0 & 0 & 0 \\
1 & -2 & -4 & 0 & 0 & 0 & 0 & 0 \\
-3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\
-4 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{matrix}
$$

where, for instance, the DC coefficient is computed as

$$\hat{T}(0,0) = \text{round}\left[\frac{T(0,0)}{Z(0,0)}\right]$$

$$= \text{round}\left[\frac{-415}{16}\right] = -26.$$

Note that the transformation and normalization process produces a large number of zero-valued coefficients. When the coefficients are reordered in accordance with the zigzag ordering pattern of Fig. 8.36(d), the resulting 1-D coefficient sequence is

$$[-26\ -3\ 1\ -3\ -2\ -6\ 2\ -4\ 1\ -4\ 1\ 1\ 5\ 0\ 2\ 0\ 0\ -1\ 2\ 0\ 0\ 0\ 0\ 0\ -1\ -1\ \text{EOB}]$$

where the EOB symbol denotes the end-of-block condition. A special EOB Huffman code word (see category 0 and run-length 0 in Table 8.19) is provided to indicate that the remainder of the coefficients in a reordered sequence are zeros.

The construction of the default JPEG code for the reordered coefficient sequence begins with the computation of the difference between the current DC coefficient and that of the previously encoded subimage. As the subimage here was taken from Fig. 8.23 and the DC coefficient of the transformed and quantized subimage to its immediate left was −17, the resulting DPCM difference is $[-26 - (-17)]$ or −9, which lies in DC difference category 4 of Table 8.17. In accordance with the default Huffman difference code of Table 8.18, the proper base code for a category 4 difference is 101 (a 3-bit code), while the total length of a completely encoded category 4 coefficient is 7 bits. The remaining 4 bits must be generated from the least significant bits (LSBs) of the difference value. For a general DC difference category (say, category K), an additional K bits are needed and computed as either the K LSBs of the positive difference or the K LSBs of the negative difference minus 1. For a difference of −9, the appropriate LSBs are $(0111) - 1$ or 0110, and the complete DPCM coded DC code word is 1010110.

The nonzero AC coefficients of the reordered array are coded similarly from Tables 8.17 and 8.19. The principal difference is that each default AC Huffman code word depends on the number of zero-valued coefficients preceding the nonzero coefficient to be coded, as well as the magnitude category of the nonzero coefficient. (See the column labeled Run/Category in Table 8.19.) Thus the first nonzero AC coefficient of the reordered array (−3) is coded as 0100. The first 2 bits of this code indicate that the coefficient was in magnitude category 2 and preceded by no zero-valued coefficients (see Table 8.17); the last 2 bits are generated by the same process used to arrive at the LSBs of the DC difference code. Continuing in this manner, the completely coded (reordered) array is

1010110 0100 001 0100 0101 100001 0110 100011 001 100011 001
001 100101 11100110 110110 0110 11110100 000 1010

where the spaces have been inserted solely for readability. Although it was not needed in this example, the default JPEG code contains a special code word for a run of 15 zeros followed by a zero (see category 0 and run-length F in Table 8.19).

The total number of bits in the completely coded reordered array (and thus the number of bits required to represent the entire 8×8, 8-bit subimage of this example) is 92. The resulting compression ratio is 512/92, or about 5.6:1.

To decompress a JPEG compressed subimage, the decoder must first recreate the normalized transform coefficients that led to the compressed bit stream. Because a Huffman-coded binary sequence is instantaneous and uniquely decodable, this step is easily accomplished in a simple lookup table manner. Here the regenerated array of quantized coefficients is

$$\begin{matrix} -26 & -3 & -6 & 2 & 2 & 0 & 0 & 0 \\ 1 & -2 & -4 & 0 & 0 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -4 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

After denormalization in accordance with Eq. (8.5-42), the array becomes

$$\begin{matrix} -416 & -33 & -60 & 32 & 48 & 0 & 0 & 0 \\ 12 & -24 & -56 & 0 & 0 & 0 & 0 & 0 \\ -42 & 13 & 80 & -24 & -40 & 0 & 0 & 0 \\ -56 & 17 & 44 & -29 & 0 & 0 & 0 & 0 \\ 18 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

where, for example, the DC coefficient is computed as

$$\dot{T}(0,0) = \hat{T}(0,0)Z(0,0) = (-26)(16) = -416.$$

The completely reconstructed subimage is obtained by taking the inverse DCT of the denormalized array in accordance with Eqs. (8.5-25) and (8.5-32) to obtain

$$\begin{matrix} -70 & -64 & -61 & -64 & -69 & -66 & -58 & -50 \\ -72 & -73 & -61 & -39 & -30 & -40 & -54 & -59 \\ -68 & -78 & -58 & -9 & 13 & -12 & -48 & -64 \\ -59 & -77 & -57 & 0 & 22 & -13 & -51 & -60 \\ -54 & -75 & -64 & 23 & 13 & 44 & 63 & 56 \\ -52 & -71 & -72 & -54 & -54 & -71 & -71 & -54 \\ -45 & -59 & -70 & -68 & -67 & -67 & -61 & -50 \\ -35 & -47 & -61 & -66 & -60 & -48 & -44 & -44 \end{matrix}$$

and level shifting each inverse transformed pixel by $+2^7$ (or +128) to yield

58	64	67	64	59	62	70	78
56	55	67	89	98	88	74	69
60	50	70	119	141	116	80	64
69	51	71	128	149	115	77	68
74	53	64	105	115	84	65	72
76	57	56	74	75	57	57	74
83	69	59	60	61	61	67	78
93	81	67	62	69	80	84	84

Any differences between the original and reconstructed subimage are a result of the lossy nature of the JPEG compression and decompression process. In this example, the errors range from −14 to +11 and are distributed as follows:

−6	−9	−6	2	11	−1	−6	−5
7	4	−1	1	11	−3	−5	3
2	9	−2	−6	−3	−12	−14	9
−6	7	0	−4	−5	−9	−7	1
−7	8	4	−1	11	4	3	−2
3	8	4	−4	2	11	1	1
2	2	5	−1	−6	0	−2	5
−6	−2	2	6	−4	−4	−6	10

The root-mean-square error of the overall compression and reconstruction process is approximately 5.9 gray levels. ■

The reconstructed subimage in the preceding example is located physically at about the center of the woman's right eye in Fig. 8.38(a). Note that both the original subimage and reconstructed result contain a local gray-level peak in the fourth row and fifth column, where a light is reflected in the woman's pupil. This local peak causes the root-mean-square error of the reconstructed subimage to exceed substantially the overall error of the completely decompressed image. In fact, it is approximately twice as great as the error associated with Fig. 8.38(a), which also was compressed with the baseline JPEG algorithm. The reason is that many of the subimages of the original image are nearly constant and can be represented with little distortion. Figure 8.38(b) provides an additional JPEG baseline compression result.

JPEG 2000

Although not yet formally adopted, JPEG 2000 extends the initial JPEG standard to provide increased flexibility in both the compression of continuous tone still images and access to the compressed data. For example, portions of a JPEG 2000 compressed image can be extracted for retransmission, storage, display,

and/or editing. The standard is based on the wavelet coding techniques of Section 8.5.3. Coefficient quantization is adapted to individual scales and subbands and the quantized coefficients are arithmetically coded on a bit-plane basis (see Section 8.4). Using the notation of the standard, an image is encoded as follows (ISO/IEC [2000]).

The first step of the encoding process is to DC level shift the samples of the *Ssiz*-bit unsigned image to be coded by subtracting 2^{Ssiz-1}. If the image has more than one *component*—like the red, green, and blue planes of a color image—each component is individually shifted. If there are exactly three components, they may be optionally decorrelated using a reversible or nonreversible linear combination of the components. The *irreversible component transform* of the standard, for example, is

$$\begin{aligned} Y_0(x, y) &= 0.299I_0(x, y) + 0.587I_1(x, y) + 0.114I_2(x, y) \\ Y_1(x, y) &= -0.16875I_0(x, y) - 0.33126I_1(x, y) + 0.5I_2(x, y) \\ Y_2(x, y) &= 0.5I_0(x, y) - 0.41869I_1(x, y) - 0.08131I_2(x, y) \end{aligned} \quad (8.6\text{-}1)$$

where I_0, I_1, and I_2 are the level-shifted input components and Y_0, Y_1, and Y_2 are the corresponding decorrelated components. If the input components are the red, green, and blue planes of a color image, Eq. (8.6-1) approximates the $R'G'B'$ to $Y'C_bC_r$ color video transform (Poynton [1996]).† The goal of the transformation is to improve compression efficiency; transformed components Y_1 and Y_2 are difference images whose histograms are highly peaked around zero.

After the image has been level shifted and optionally decorrelated, its components are optionally divided into *tiles*. Tiles are rectangular arrays of pixels that contain the same relative portion of all components. Thus, the tiling process creates *tile components* that can be extracted and reconstructed independently, providing a simple mechanism for accessing and/or manipulating a limited region of a coded image.

The one-dimensional discrete wavelet transform of the rows and columns of each tile component is then computed. For error-free compression, the transform is based on a biorthogonal, 5-3 coefficient scaling and wavelet vector (Le Gall and Tabatabai [1988]). A rounding procedure is defined for non-integer-valued transform coefficients. In lossy applications, a 9-7 coefficient scaling-wavelet vector (Antonini, Barlaud, Mathieu, and Daubechies [1992]) is employed. In either case, the transform is computed using the fast wavelet transform of Section 7.4 or via a *lifting-based* approach (Mallat [1999]). The coefficients needed to construct a 9-7 FWT analysis filter bank are given in Table 8.20. The complementary lifting-based implementation involves six sequential "lifting" and "scaling" operations:

---

† $R'G'B'$ is a gamma corrected, nonlinear version of a linear CIE (International Commission on Illumination) RGB colorimetry value. Y' is luminance and C_b and C_r are color differences (i.e., scaled $B' - Y'$ and $R' - Y'$ values).

Filter Tap	Highpass Wavelet Coefficient	Lowpass Scaling Coefficient
0	−1.115087052456994	0.6029490182363579
±1	0.5912717631142470	0.2668641184428723
±2	0.05754352622849957	−0.07822326652898785
±3	−0.09127176311424948	−0.01686411844287495
±4	0	0.02674875741080976

TABLE 8.20 Impulse responses of the low and highpass analysis filters for an irreversible 9-7 wavelet transform.

$$
\begin{aligned}
&Y(2n+1) = X(2n+1) + \alpha[X(2n) + X(2n+2)], \\
&\qquad\qquad i_0 - 3 \le 2n + 1 < i_1 + 3 \\
&Y(2n) = X(2n) + \beta[Y(2n-1) + Y(2n+1)], \quad i_0 - 2 \le 2n < i_1 + 2 \\
&Y(2n+1) = Y(2n+1) + \gamma[Y(2n) + Y(2n+2)], \\
&\qquad\qquad i_0 - 1 \le 2n + 1 < i_1 + 1 \qquad (8.6\text{-}2) \\
&Y(2n) = Y(2n) + \delta[Y(2n-1) + Y(2n+1)], \quad i_0 \le 2n < i_1 \\
&Y(2n+1) = -K \cdot Y(2n+1), \quad i_0 \le 2n + 1 < i_1 \\
&Y(2n) = Y(2n)/K, \quad i_0 \le 2n < i_1.
\end{aligned}
$$

Here, X is the tile component being transformed, Y is the resulting transform, and i_0 and i_1 define the position of the tile component within a component. That is, they are the indices of the first sample of the tile-component row or column being transformed and the one immediately following the last sample. Variable n assumes values based on i_0, i_1, and which of the six operations is being performed. If $n < i_0$ or $n \ge i_1$, $X(n)$ is obtained by symmetrically extending X. For example, $X(i_0 - 1) = X(i_0 + 1)$, $X(i_0 - 2) = X(i_0 + 2)$, $X(i_1) = X(i_1 - 2)$, and $X(i_1 + 1) = X(i_1 - 3)$. At the conclusion of the lifting and scaling operations, the even-indexed values of Y are equivalent to the FWT lowpass filtered output; the odd-indexed values of Y correspond to the highpass FWT filtered result. Lifting parameters α, β, γ, and δ are −1.586134342, −0.052980118, 0.882911075, and 0.433506852, respectively. Scaling factor K is 1.230174105.

The transformation just described produces four subbands—a low-resolution approximation of the tile component and the component's horizontal, vertical, and diagonal frequency characteristics. Repeating the transformation N_L times, with subsequent iterations restricted to the previous decomposition's approximation coefficients, produces an N_L-scale wavelet transform. Adjacent scales are related spatially by powers of 2 and the lowest scale contains the only explicitly defined approximation of the original tile component. As can be surmised from Fig. 8.46, where the notation of the standard is summarized for the case of $N_L = 2$, a general N_L-scale transform contains $3N_L + 1$ subbands whose coefficients are denoted a_b, for $b = N_L LL, N_L HL, \ldots, 1HL, 1LH, 1HH$. The standard does not specify the number of scales to be computed.

When each of the tile components has been processed, the total number of transform coefficients is equal to the number of samples in the original image—

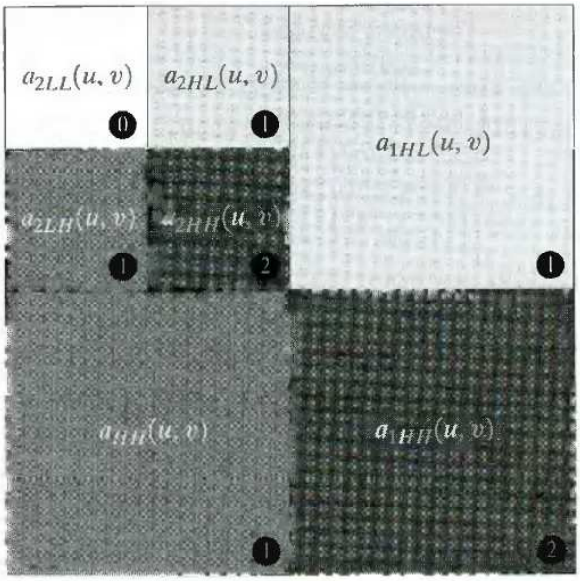


FIGURE 8.46 JPEG 2000 two-scale wavelet transform tile-component coefficient notation and analysis gain.

but the important visual information is concentrated in a few coefficients. To reduce the number of bits needed to represent the transform, coefficient $a_b(u, v)$ of subband b is quantized to value $q_b(u, v)$ using

$$q_b(u, v) = \text{sign}[a_b(u, v)] \cdot \text{floor}\left[\frac{|a_b(u, v)|}{\Delta_b}\right] \tag{8.6-3}$$

where *quantiztion step size* Δ_b is

$$\Delta_b = 2^{R_b - \varepsilon_b}\left(1 + \frac{\mu_b}{2^{11}}\right), \tag{8.6-4}$$

R_b is the *nominal dynamic range* of subband b, and ε_b and μ_b are the number of bits allotted to the *exponent* and *mantissa* of the subband's coefficients. The nominal dynamic range of subband b is the sum of the number of bits used to represent the original image and the *analysis gain* bits for subband b. Subband analysis gain bits follow the simple pattern shown in Fig. 8.46. For example, there are two analysis gain bits for subband $b = 1HH$.

For error-free compression, $\mu_b = 0$, $R_b = \varepsilon_b$, and $\Delta_b = 1$. For irreversible compression, no particular quantization step size is specified in the standard. Instead, the number of exponent and mantissa bits must be provided to the decoder on a subband basis, called *explicit quantization*, or for the $N_L LL$ subband only, called *implicit quantization*. In the latter case, the remaining subbands are quantized using extrapolated $N_L LL$ subband parameters. Letting ε_0 and μ_0 be the number of bits allocated to the $N_L LL$ subband, the extrapolated parameters for subband b are

$$\begin{aligned} \mu_b &= \mu_0 \\ \varepsilon_b &= \varepsilon_0 + nsd_b - nsd_0 \end{aligned} \tag{8.6-5}$$

where nsd_b denotes the number of subband decomposition levels from the original image tile component to subband b.

The final steps of the encoding process are *coefficient bit modeling*, arithmetic coding, bit-stream layering, and packetizing. The coefficients of each transformed tile-component's subbands are arranged into rectangular blocks called *code blocks*, which are individually coded a bit plane at a time. Starting from the most significant bit plane with a nonzero element, each bit plane is processed in three passes. Each bit of a bit plane is coded in only one of the three passes, which are called *significance propagation*, *magnitude refinement*, and *cleanup*. The outputs are then arithmetically coded and grouped with similar passes from other code blocks to form *layers*. A layer is an arbitrary number of groupings of coding passes from each code block. The resulting layers are finally partitioned into *packets*, providing an additional method of extracting a spatial region of interest from the total code stream. Packets are the fundamental unit of the encoded code stream.

JPEG 2000 decoders simply invert the operations described previously. After decoding the bit modeled, arithmetically coded, layered, and packetized codestream, a user-selected number of the original image's tile-component subbands are reconstructed. Although the encoder may have encoded M_b bit planes for a particular subband, the user—due to the embedded nature of the code stream—may choose to decode only N_b bit planes. This amounts to quantizing the code block's coefficients using a step size of $2^{M_b - N_b} \cdot \Delta_b$. Any nondecoded bits are set to zero and the resulting coefficients, denoted $\bar{q}_b(u, v)$, are dequantized using

$$R_{q_b}(u, v) = \begin{cases} \left(\bar{q}_b(u, v) + 2^{M_b - N_b(u,v)}\right) \cdot \Delta_b & \bar{q}_b(u, v) > 0 \\ \left(\bar{q}_b(u, v) - 2^{M_b - N_b(u,v)}\right) \cdot \Delta_b & \bar{q}_b(u, v) < 0 \\ 0 & \bar{q}_b(u, v) = 0 \end{cases} \tag{8.6-6}$$

where $R_{q_b}(u, v)$ denotes a dequantized transform coefficient and $N_b(u, v)$ is the number of decoded bit planes for $\bar{q}_b(u, v)$. The dequantized coefficients are then inverse transformed by column and by row using an FWT^{-1} filter bank whose coefficients are obtained from Table 8.20 and Eq. (7.1-15) or via the following lifting-based operations:

$$\begin{aligned} X(2n) &= K \cdot Y(2n), \quad i_0 - 3 \le 2n < i_1 + 3 \\ X(2n+1) &= (-1/K) \cdot Y(2n+1), \quad i_0 - 2 \le 2n + 1 < i_1 + 2 \\ X(2n) &= X(2n) - \delta[X(2n-1) + X(2n+1)], \\ &\qquad i_0 - 3 \le 2n < i_1 + 3 \\ X(2n+1) &= X(2n+1) - \gamma[X(2n) + X(2n+2)], \\ &\qquad i_0 - 2 < 2n + 1 < i_1 + 2 \\ X(2n) &= X(2n) - \beta[X(2n-1) + X(2n+1)], \quad i_0 - 1 \le 2n < i_1 + 1 \\ X(2n+1) &= X(2n+1) - \alpha[X(2n) + X(2n+2)], \quad i_0 \le 2n + 1 < i_1 \end{aligned} \tag{8.6-7}$$

where parameters $\alpha, \beta, \gamma, \delta$, and K are as defined for Eq. (8.6-2). Dequantized coefficient row or column element $Y(n)$ is symmetrically extended when necessary. The final decoding steps are the assembly of the component tiles, inverse

component transformation (if required), and DC level shifting. For irreversible coding, the inverse component transformation is

$$\begin{aligned} I_0(x, y) &= Y_0(x, y) + 1.402Y_2(x, y) \\ I_1(x, y) &= Y_0(x, y) - 0.34413Y_1(x, y) - 0.71414Y_2(x, y) \\ I_2(x, y) &= Y_0(x, y) + 1.772Y_1(x, y) \end{aligned} \tag{8.6-8}$$

and the transformed pixels are shifted by $+2^{Ssiz-1}$. Figures 8.40 and 8.41 of Section 8.5.3, which illustrate compression ratios from 34:1 to 167:1, were generated using the irreversible JPEG 2000 algorithm.

8.6.3 Video Compression Standards

Video compression standards extend the transform-based, still image compression techniques of the previous section to include methods for reducing temporal or frame-to-frame redundancies. Although there are a variety of video coding standards in use today, most rely on similar video compression techniques. Depending on the intended application, the standards can be grouped into two broad categories: (1) video teleconferencing standards and (2) multimedia standards.

A number of video teleconferencing standards, including H.261 (also referred to as PX64), H.262, H.263, and H.320, have been defined by the International Telecommunications Union (ITU), the successor to the CCITT. H.261 is intended for operation at affordable telecom bit rates and to support full motion video transmission over T1[†] lines with delays of less than 150 ms. Delays exceeding 150 ms do not provide viewers the "feeling" of direct visual feedback. H.263, on the other hand, is designed for very low bit rate video, in the range of 10 to 30 kbit/s, and H.320, a superset of H.261, is constructed for Integrated Services Digital Network[‡] (ISDN) bandwidths. Each standard uses a motion-compensated, DCT-based coding scheme. Since motion estimation is difficult to perform in the transform domain, blocks of pixels, called *macroblocks*, are compared to neighboring blocks of the previous frame and used to compute a motion compensated prediction error. The prediction error is then discrete cosine transformed in 8×8 pixel blocks, quantized, and coded for transmission or storage.

Multimedia video compression standards for video on demand, digital HDTV broadcasting, and image/video database services use similar motion estimation and coding techniques. The principal standards—MPEG-1, MPEG-2, and MPEG-4—were developed under the auspices of the Motion Picture Experts Group of the CCITT and ISO. MPEG-1 is an "entertainment quality" coding standard for the storage and retrieval of video on digital media like compact disk read-only memories (CD-ROMs). It supports bit rates on the order of

[†] The T1 line was introduced by the Bell system for digital voice communications over short distances of 10 to 50 miles. Twenty-four telephone channels are time-multiplexed, sampled, and coded into a 1.544 Mbit/s PCM (pulse code modulation) signal for transmission over a single T1 line.

[‡] Two ISDN "B" channels provide sufficient bandwidth (i.e., 128 kbit/s) to transmit compressed 320×240 images at 15 frames per second.

1.5 Mbit/s. MPEG-2 addresses applications involving video quality between NTSC/PAL[†] and CCIR 601[‡]; bit rates from 2 to 10 Mbit/s, a range that is suitable for cable TV distribution and narrow-channel satellite broadcasting, are supported. The goal of both MPEG-1 and MPEG-2 is to make the storage and transmission of digital audio and video (AV) material efficient. MPEG-4, on the other hand, provides (1) improved video compression efficiency; (2) content-based interactivity, such as AV object-based access and efficient integration of natural and synthetic data; and (3) universal access, including increased robustness in error-prone environments, the ability to add or drop AV objects, and object resolution scalability. Although these functionalities create a need to segment arbitrarily shaped video objects, segmentation is not a part of the standard. A great deal of video content—like computer games—is produced and readily available in the form of video objects. MPEG-4 targets bit rates between 5 and 64 kbit/s for mobile and public switched telephone network (PSTN) applications and up to 4 Mbit/s for TV and film applications. In addition, it supports both constant bit rate and variable bit rate coding.

Like the ITU teleconferencing standards, MPEG standards are built around a hybrid block-based DPCM/DCT coding scheme. Figure 8.47 shows a typical MPEG encoder. It exploits redundancies within and between adjacent video frames, motion uniformity between frames, and the psychovisual properties of the human visual system. The input of the encoder is an 8×8 array of pixels, called an *image block*. The standards define a *macroblock* as a 2×2 array of image blocks (i.e., a 16×16 array of image elements) and a *slice* as a row of nonoverlapping macroblocks. For color video, a macroblock, is composed of four luminance blocks, denoted Y_1 through Y_4, and two chrominance blocks, C_b and C_r. Recall that color difference signal C_b is blue minus luminance and C_r is red minus luminance. Because the eye has far less spatial acuity for color than for luminance, these two components are often sampled at half the horizontal and vertical resolution of the luminance signal, resulting in a 4 : 1 : 1 sampling ratio between $Y' : C_b : C_r$.

The grayed elements of the primary input-to-output path in Fig. 8.47 parallel the transformation, quantization, and variable-length coding operations of a JPEG encoder. The principal difference is the input, which may be a conventional block of image data or the difference between a conventional block and a prediction of it based on similar blocks in previous and/or subsequent video frames. This leads to three basic types of encoded output frames:

1. *Intraframe* or *independent frame* (*I-frame*). An I-frame is compressed independently of all previous and future video frames. Of the three possible encoded output frames, it most highly resembles a JPEG encoded image. Moreover, it is the reference point for the motion estimation needed to generate subsequent *P*- and *B-frames*. I-frames provide the highest degree of

[†]NTSC and PAL are acronyms for the National Television System Committee and Phase Alternate Line, respectively. Both are composite color video standards.

[‡]CCIR is an acronym for the International Radio Consultive Committee.

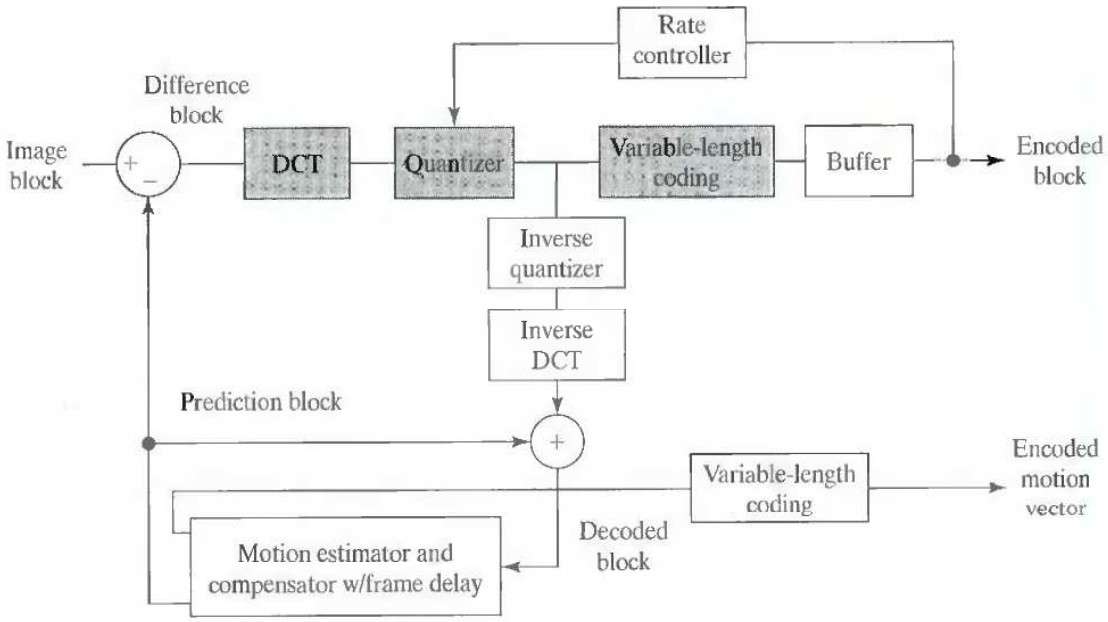


FIGURE 8.47 A basic DPCM/DCT encoder for motion compensated video compression.

random access, ease of editing, and greatest resistance to the propagation of transmission error. As a result, all standards require their periodic insertion into the compressed codestream.

2. *Predictive frame* (*P-frame*). A P-frame is the compressed difference between the current frame and a prediction of it based on the previous I- or P-frame. The difference is formed in the leftmost summer of Fig. 8.47. The prediction is motion compensated and typically involves sliding the decoded block in the lower part of Fig. 8.47 around its immediate neighborhood in the current frame and computing a measure of correlation (such as the sum of the square of the pixel-by-pixel differences). In fact, the process is often carried out in subpixel increments (such as sliding the subimage $\frac{1}{4}$ pixels at a time), which necessitates interpolating pixel values prior to computing the correlation measure. The computed motion vector is variable-length coded and transmitted as an integral part of the encoded data stream. Motion estimation is carried out on the macroblock level.
3. *Bidirectional frame* (*B-frame*). A B-frame is the compressed difference between the current frame and a prediction of it based on the previous I- or P-frame and next P-frame. Accordingly, the decoder must have access to both past and future reference frames. The encoded frames are therefore reordered before transmission; the decoder reconstructs and displays them in the proper sequence.

The encoder of Fig. 8.47 is designed to generate a bit stream that matches the capacity of the intended video channel. To accomplish this, the quantization parameters are adjusted by the *rate controller* as a function of the occupancy of the output *buffer*. As the buffer becomes fuller, the quantization is made coarser, so that fewer bits stream into the buffer.

Summary

The principal objectives of this chapter were to present the theoretic foundation of digital image compression and to describe the most commonly used compression methods that form the core of the technology as it exists currently. Although the level of the presentation is introductory in nature, the depth and breadth of the material covered are sufficient to serve as the basis for independent reading in this subject area. The references provide an entry into the extensive body of literature dealing with image compression and related topics. In addition to extensive uses involving gray-scale imagery, compression methods are playing an increasingly important role in document image storage and transmission, as evidenced by the emergence of the international standards discussed in Section 8.6. In addition to medical imaging, compression is one of the few areas of image processing that has received a sufficiently broad commercial appeal to warrant the adoption of widely accepted standards.

References and Further Reading

The introductory material of the chapter, which is generally confined to Sections 8.1 and 8.2, is basic to image compression and may be found in one form or another in most of the general image processing books cited at the end of Chapter 1. The material in Section 8.1.3 on improved gray-scale quantization is based on Bisignani, Richards, and Whelan [1966]. For additional information on the human visual system, see Netravali and Limb [1980], as well as in Huang [1966], Schreiber and Knapp [1958], and the references cited at the end of Chapter 2. Subjective fidelity criteria are discussed in Frendendall and Behrend [1960]. Error detecting and correcting codes are covered in most introductory texts on switching or finite automata theory, as well as in general information theory texts.

The material in Section 8.3 is based on several excellent books on information theory. Noteworthy are Abramson [1963], Blahut [1987], and Berger [1971]. Shannon's classic paper, "A Mathematical Theory of Communication" [1948], lays the foundation for most of the material in the section and is another excellent reference.

The descriptions of the error-free encoding techniques of Section 8.4 are, for the most part, based on the original papers cited in the text or as follows. The algorithms covered are representative of the work in this area, but are by no means exhaustive. The material on LZW coding has its origins in the work of Ziv and Lempel [1977, 1978]. The material on arthmetic coding follows the development in Witten, Neal, and Cleary [1987]. One of the more important implementations of arithmetic coding is summarized in Pennebaker et al. [1988]. For additional information on bit-plane coding, see Schwartz and Barker [1966] and the tutorial by Rabbani and Jones [1991], which also contains a good discussion of lossless predictive coding. Huang and Hussian [1975] first published the details of white-block skipping. Relative address coding and predictive differential quantizing were first reported by Yamazaki, Wakahara, and Teramura [1976] and Huang and Tretiak [1972], respectively. The adaptive predictor of Eq. (8.5-19) is from Graham [1958].

The material in Section 8.5 covers the principal lossy encoding approaches. Various other methods are directly based on these techniques. Noteworthy among them are *hybrid encoding* (Habibi [1974]), a scheme that combines 1-D transform coding and DPCM to obtain about the same performance as 2-D transform coding using fewer computations; *subband coding* (Woods and O'Neil [1986]), in which an image is filtered into a set of images (with different spatial frequencies) that may be individually DPCM coded; and *interframe coding* (Roese et al. [1977]), where the redundancy between successive frames in a time sequence of images is reduced by using a predictive or transform coding

approach. In addition, a variety of lossy techniques are closely related to the techniques described. These include, among others, *block truncation coding* (Delp and Mitchell [1979]), in which a 1-bit quantizer is designed for each $n \times n$ block of a subdivided image; *vector quantization* (Linde et al. [1980]), in which an image is decomposed into vectors (containing pixels, transform coefficients, and so on) that are matched against a code book of possible vectors and coded to indicate the best fit; and *hierarchical coding* (Knowlton [1980]), which usually involves the generation of a pyramid-structured data set that can be progressively accessed to obtain better and better representations of the original image. These references do not necessarily cite the inventor of the techniques; they provide a starting point for additional reading on the methods. Other articles or books of interest include Tasto and Wintz [1971], Gharavi and Tabatabai [1988], Baylon and Lim [1990], Candy et al. [1971], Jain and Jain [1981], Healy and Mitchell [1981], Lema and Mitchell [1984], Udpikar and Raina [1987], Gray [1984], Equitz [1989], Sezan et al. [1989], Tanimoto [1979], Blume and Fand [1989], Rabbani and Jones [1991], and Storer and Reif [1991]. Almost every issue of the *IEEE Transactions on Image Processing* includes several articles on video and still image compression, with many of the articles related to wavelet- and fractal-based compression, vector quantization, and video motion compensation. See, for example, Boulgouris et al. [2001], Martin and Bell [2001], Chen and Wilson [2000], Hartenstein et al. [2000], Yang and Ramchandran [2000], and Meyer et al. [2000] as a starting point for further reading and references.

Section 8.6 is based primarily on the published drafts and formal standards of the International Standards Organization and the Consultative Committee of International Telephone and Telegraph. These documents are available from the standards organizations or the American National Standards Institute (ANSI). Additional references on compression standards include Hunter and Robinson [1980], Ang et al. [1991], Fox [1991], Pennebaker and Mitchell [1992], Bhatt et al. [1997], Sikora [1997], Bhaskaran and Konstantinos [1997], Ngan et al. [1999], Weinberger et al. [2000], and Symes [2001].

Several survey articles have been devoted to the field of image compression. Noteworthy are Netravali and Limb [1980], A. K. Jain [1981], a special issue on picture communication systems in the *IEEE Transactions on Communications* [1981], a special issue on the encoding of graphics in the *Proceedings of IEEE* [1980], a special issue on visual communication systems in the *Proceedings of the IEEE* [1985], a special issue on image sequence compression in the *IEEE Transactions on Image Processing* [1994], and a special issue on vector quantization in the *IEEE Transactions on Image Processing* [1996].

Problems

See inside front cover

Detailed solutions to the problems marked with a star can be found in the book web site. The site also contains suggested projects based on the material in this chapter.

8.1 **(a)** Can variable-length coding procedures be used to compress a histogram equalized image with 2^n gray levels? Explain.

(b) Can such an image contain interpixel redundancies that could be exploited for data compression?

8.2 One variation of the run-length coding procedure described in Section 8.1.2 involves (1) coding only the runs of 0's or 1's (not both) and (2) assigning a special code to the start of each line to reduce the effect of transmission errors. One possible code pair is (x_k, r_k), where x_k and r_k represent the kth run's starting coordinate and run length, respectively. The code $(0, 0)$ is used to signal each new line.

(a) Derive a general expression for the maximum average runs per scan line required to guarantee data compression when run-length coding a $2^n \times 2^n$ binary image.

(b) Compute the maximum allowable value for $n = 10$.

★**8.3** Consider an 8-pixel line of gray-scale data, $\{12, 12, 13, 13, 10, 13, 57, 54\}$, which has been uniformly quantized with 6-bit accuracy. Construct its 3-bit IGS code.

8.4 Compute the rms error and rms signal-to-noise ratios for the decoded IGS data of Problem 8.3.

8.5 **(a)** Use the Hamming (7, 4) code to code the IGS quantized data of Table 8.2.

★ **(b)** Determine which bit, if any, is in error in the Hamming encoded messages 1100111, 1100110, and 1100010. What are the decoded values?

★**8.6** The base e unit of information is commonly called a *nat*, and the base 10 information unit is called a *Hartley*. Compute the conversion factors needed to relate these units to the base 2 unit of information (the bit).

★**8.7** Prove that, for a zero-memory source with q symbols, the maximum value of the entropy is $\log q$, which is achieved if and only if all source symbols are equiprobable. *Hint*: Consider the quantity $\log q - H(\mathbf{z})$ and note the inequality $\ln x \leq x - 1$.

8.8 Calculate the various probabilities associated with the information channel in which $A = \{0, 1\}$, $B = \{0, 1\}$, $\mathbf{z} = [0.75, 0.25]^T$ and

$$\mathbf{Q} = \begin{bmatrix} \frac{2}{3} & \frac{1}{3} \\ \frac{1}{10} & \frac{9}{10} \end{bmatrix}.$$

Include $P(a = 0)$, $P(a = 1)$, $P(b = 0)$, $P(b = 1)$, $P(b = 0|a = 0)$, $P(b = 0|a = 1)$, $P(b = 1|a = 0)$, $P(b = 1|a = 1)$, $P(a = 0|b = 0)$, $P(a = 0|b = 1)$, $P(a = 1|b = 0)$, $P(a = 1|b = 1)$, $P(a = 0, b = 0)$, $P(a = 0, b = 1)$, $P(a = 1, b = 0)$, and $P(a = 1, b = 1)$.

★**8.9** Consider the binary information source and BSC of the example in Section 8.3.2 and let $p_{bs} = 3/4$ and $p_e = 1/3$.

(a) What is the entropy of the source?

(b) How much less uncertainty about the input is there when the output has been observed?

(c) What is this difference in uncertainty called and how does it compare numerically to the channel's capacity?

8.10 A *binary erasure channel* is one in which there is a finite probability β that a transmitted symbol will not be received. The channel has three possible outputs: a 0, an erasure (no received symbol), and a 1. These three outcomes form the three rows of the binary erasure channel matrix

$$\mathbf{Q} = \begin{bmatrix} 1 - \beta & 0 \\ \beta & \beta \\ 0 & 1 - \beta \end{bmatrix}.$$

(a) Find the capacity of the channel.

★ **(b)** Would you prefer a binary symmetric channel with a 0.125 probability of error or an erasure channel with probability of erasure $\beta = 0.5$?

8.11 The rate distortion function of a zero-memory Gaussian source of arbitrary mean and variance σ^2 with respect to the mean-square error criterion (Berger [1971]) is

$$R(D) = \begin{cases} \frac{1}{2}\log\frac{\sigma^2}{D} & \text{for } 0 \le D \le \sigma^2 \\ 0 & \text{for } D \ge \sigma^2. \end{cases}$$

★ **(a)** Plot this function.

(b) What is $D_{\max}$?

(c) If a distortion of no more than 75% of the source's variance is allowed, what is the maximum compression that can be achieved?

8.12 **(a)** How many unique Huffman codes are there for a three-symbol source?

(b) Construct them.

8.13 **(a)** Compute the entropy of the source whose symbol probabilities are defined in Table 8.1.

(b) Construct a Huffman code for the source symbols and explain any differences between the constructed code and Code 2 of the table.

(c) Construct the best B_1-code for this distribution.

(d) Construct the best 2-bit binary shift code.

(e) Divide the symbols into two blocks of four and construct the best Huffman shift code.

(f) Compute the average word lengths for each code and compare them to the entropy from part (a).

★ **8.14** The arithmetic decoding process is the reverse of the encoding procedure. Decode the message 0.23355 given the coding model

Symbol	Probability
a	0.2
e	0.3
i	0.1
o	0.2
u	0.1
!	0.1

8.15 Use the LZW coding algorithm of Section 8.4.2 to encode the 7-bit ASCII string "aaaaaaaaaaa".

★ **8.16** Devise an algorithm for decoding the LZW encoded output of Example 8.12. Since the dictionary that was used during the encoding is not available, the code book must be reproduced as the output is decoded.

8.17 **(a)** Construct the entire 4-bit Gray code.

(b) Create a general procedure for converting a Gray-coded number to its binary equivalent and use it to decode 0111010100111.

8.18 A 64×64 pixel binary image has been coded using 1-D WBS with blocks of four pixels. The WBS code for one line of the image was 0110010000001000010010000000, where a 0 is used to represent a black pixel.

(a) Decode the line.

(b) Create a 1-D iterative WBS procedure that begins by looking for all white lines (a 64-pixel block) and successively halves nonwhite intervals until four pixel blocks are reached.

(c) Use your algorithm to code the previously decoded line. It should require fewer bits.

8.19 ★ (a) Explain why the first similar transition past e on the previous line is used as c' in relative address coding.

(b) Can you devise an alternate approach?

8.20 An image whose autocorrelation function is of the form of Eq. (8.5-12) with $\rho_h = 0$ is to be DPCM coded using a second-order predictor.

(a) Form the autocorrelation matrix $\mathbf{R}$ and vector $\mathbf{r}$.

(b) Find the optimal prediction coefficients.

(c) Compute the variance of the prediction error that would result from using the optimal coefficients.

★8.21 Derive the Lloyd-Max decision and reconstruction levels for $L = 4$ and the uniform probability density function

$$p(s) = \begin{cases} \dfrac{1}{2A} & -A \le s \le A \\ 0 & \text{otherwise.} \end{cases}$$

8.22 Use the CCITT Group 4 compression algorithm to code the second line of the following two-line segment:

0	1	1	0	0	1	1	1	0	0	1	1	1	1	1	1	1	1	0	0	0	0	1
1	1	1	1	1	1	1	0	0	0	1	1	1	0	0	0	0	1	1	1	1	1	1

Assume that the initial reference element a_0 is located on the first pixel of the second line segment.

★8.23 (a) List all the members of the JPEG DC coefficient difference category 3.

(b) Compute their default Huffman codes using Table 8.18.

8.24 A radiologist from a well-known research hospital recently attended a medical conference at which a system that could transmit 4096×4096 12-bit digitized X-ray images over standard T1 phone lines was exhibited. The system transmitted the images in a compressed form using a *progressive* technique in which a reasonably good approximation of the X-ray was first reconstructed at the viewing station and then refined gradually to produce an error-free display. The transmission of the data needed to generate the first approximation took approximately 5 or 6 s. Refinements were made every 5 or 6 s (on the average) for the next 1 min, with the first and last refinements having the most and least significant impact on the reconstructed X-ray, respectively. The physician was favorably impressed with the system, because she could begin her diagnosis by using the first approximation of the X-ray and complete it as the error-free reconstruction of the X-ray was being generated. Upon returning to her office, she submitted a purchase request to the hospital administrator. Unfortunately, the hospital was on a relatively tight budget, which recently had been stretched thinner by the hiring of an aspiring young electrical engineering graduate. To appease the radiologist, the administrator gave the young engineer the task of designing such a system. (He thought it might be

cheaper to design and build a similar system in-house. The hospital currently owned some of the elements of such a system, but the transmission of the raw X-ray data took more than 2 min.) The administrator asked the engineer to have an initial block diagram by the afternoon staff meeting. With little time and only a copy of *Digital Image Processing* (this text, of course) from his recent school days in hand, the engineer was able to devise conceptually a system to satisfy the transmission and associated compression requirements. Construct a conceptual block diagram of such a system, specifying the compression techniques you would recommend.

8.25 Show that the lifting-based wavelet transform defined by Eq. (8.6-2) is equivalent to the traditional FWT filter bank implementation using the coefficients in Table 8.20. Define the filter coefficients in terms of $\alpha, \beta, \gamma, \delta$, and K.

8.26 Compute the quantization step sizes of the subbands for a JPEG 2000 encoded image in which implicit quantization is used and 8 bits are allotted to the mantissa and exponent of the $2LL$ subband.

★8.27 Draw the block diagram of the companion MPEG decoder for the encoder in Fig. 8.47.

9 Morphological Image Processing

In form and feature, face and limb,
I grew so like my brother
That folks got taking me for him
And each for one another.

Henry Sambrooke Leigh, Carols of Cockayne, The Twins

Preview

The word *morphology* commonly denotes a branch of biology that deals with the form and structure of animals and plants. We use the same word here in the context of *mathematical morphology* as a tool for extracting image components that are useful in the representation and description of region shape, such as boundaries, skeletons, and the convex hull. We are interested also in morphological techniques for pre- or postprocessing, such as morphological filtering, thinning, and pruning.

The language of mathematical morphology is set theory. As such, morphology offers a unified and powerful approach to numerous image processing problems. Sets in mathematical morphology represent objects in an image. For example, the set of all black pixels in a binary image is a complete morphological description of the image. In binary images, the sets in question are members of the 2-D integer space Z^2 (see Section 2.4.2), where each element of a set is a tuple (2-D vector) whose coordinates are the (x, y) coordinates of a black (or white, depending on convention) pixel in the image. Gray-scale digital images can be represented as sets whose components are in Z^3. In this case, two components of each element of the set refer to the coordinates of a pixel, and the third corresponds to its discrete gray-level value. Sets in higher dimensional spaces can contain other image attributes, such as color and time varying components.

In the following sections we develop and illustrate several important concepts in mathematical morphology. Many of the ideas introduced here can be formulated in terms of n-dimensional Euclidean space, E^n. However, our interest

initially is on binary images whose components are elements of Z^2. We discuss extensions to gray-scale images in Section 9.6.

The material in this chapter begins a transition from a focus on purely image processing methods, whose input and output are images, to processes in which the inputs are images, but the outputs are attributes extracted from those images, in the sense defined in Section 1.1. Tools like morphology and related concepts are a cornerstone of the mathematical foundation that is utilized for extracting "meaning" from an image. Other approaches are developed and applied in the remaining chapters of this book.

9.1 Preliminaries

In this section we introduce some basic concepts from set theory that are needed as foundation for the remaining sections of this chapter.

9.1.1 Some Basic Concepts from Set Theory

Let A be a set in Z^2. If $a = (a_1, a_2)$ is an element of A, then we write

$$a \in A. \tag{9.1-1}$$

Similarly, if a is not an element of A, we write

$$a \notin A. \tag{9.1-2}$$

The set with no elements is called the *null* or *empty set* and is denoted by the symbol $\varnothing$.

A set is specified by the contents of two braces: $\{\cdot\}$. The elements of the sets with which we are concerned in this chapter are the *coordinates* of pixels representing objects or other features of interest in an image. For example, when we write an expression of the form $C = \{w | w = -d, \text{ for } d \in D\}$ we mean that set C is the set of elements, w, such that w is formed by multiplying each of the two coordinates of all the elements of set D by -1.

If every element of a set A is also an element of another set B, then A is said to be a *subset* of B, denoted as

$$A \subseteq B. \tag{9.1-3}$$

The *union* of two sets A and B, denoted by

$$C = A \cup B, \tag{9.1-4}$$

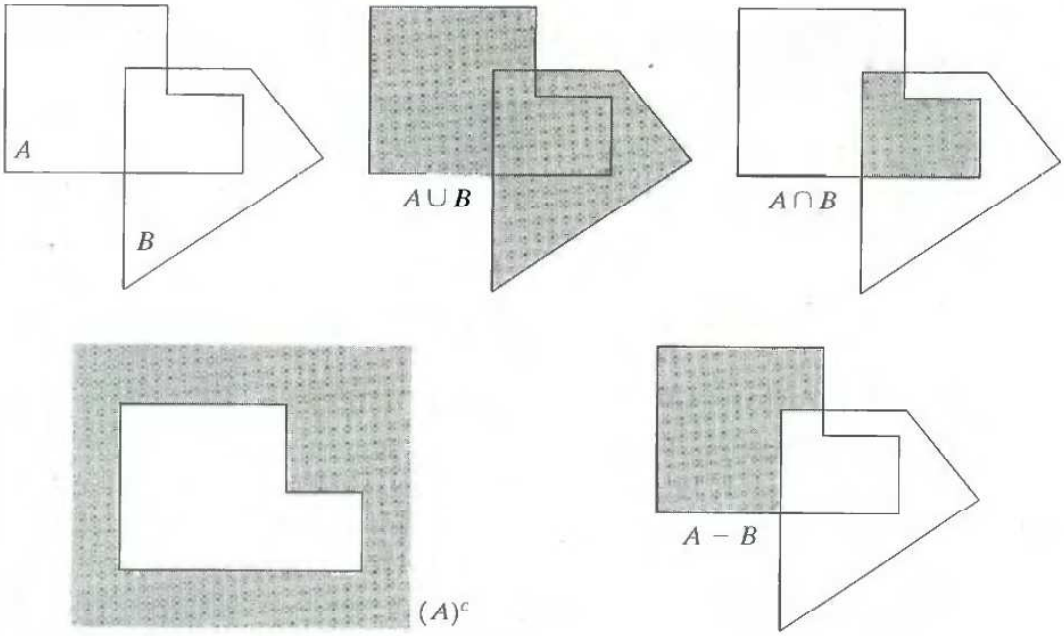
is the set of all elements belonging to either A, B, or both. Similarly, the *intersection* of two sets A and B, denoted by

$$D = A \cap B, \tag{9.1-5}$$

is the set of all elements belonging to both A and B.

Two sets A and B are said to be *disjoint* or *mutually exclusive* if they have no common elements. In this case,

$$A \cap B = \varnothing. \tag{9.1-6}$$



a b c
d e

FIGURE 9.1 (a) Two sets A and B. (b) The union of A and B. (c) The intersection of A and B. (d) The complement of A. (e) The difference between A and B.

The *complement* of a set A is the set of elements not contained in A:

$$A^c = \{w \mid w \notin A\}. \tag{9.1-7}$$

The *difference* of two sets A and B, denoted $A - B$, is defined as

$$A - B = \{w \mid w \in A, w \notin B\} = A \cap B^c. \tag{9.1-8}$$

We see that this is the set of elements that belong to A, but not to B. Figure 9.1 illustrates the preceding concepts. The result of the set operation indicated in each figure is shown in gray.

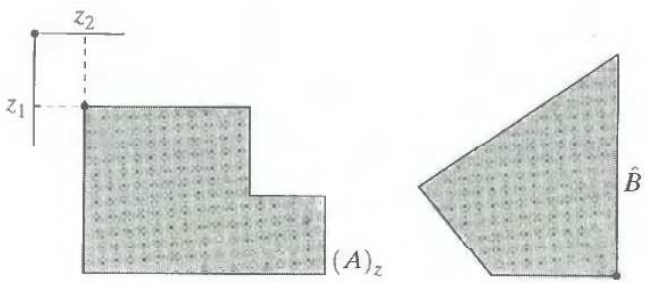
We need two additional definitions that are used extensively in morphology but generally are not found in basic texts on set theory. The *reflection* of set B, denoted $\hat{B}$, is defined as

$$\hat{B} = \{w \mid w = -b, \quad \text{for} \quad b \in B\}. \tag{9.1-9}$$

The *translation* of set A by point $z = (z_1, z_2)$, denoted $(A)_z$, is defined as

$$(A)_z = \{c \mid c = a + z, \quad \text{for} \quad a \in A\}. \tag{9.1-10}$$

Figure 9.2 illustrates these two definitions using the sets from Fig. 9.1. The black dot identifies the origin of the sets shown in the figure.



a b

FIGURE 9.2 (a) Translation of A by z. (b) Reflection of B. The sets A and B are from Fig. 9.1.

TABLE 9.1
The three basic logical operations.

p	q	p AND q (also $p \cdot q$)	p OR q (also $p + q$)	NOT (p) (also p)
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

9.1.2 Logic Operations Involving Binary Images

The majority of applications based on the morphological concepts discussed in this chapter involve binary images. Logic operations, although simple in nature, provide a powerful complement to implementation of image processing algorithms based on morphology. Logic operations were introduced in Section 3.4 in connection with masking. In the following discussion, we are interested in logic operations involving binary pixels and images.

The principal logic operations used in image processing are AND, OR, and NOT (COMPLEMENT). Their properties are summarized in Table 9.1. These operations are *functionally complete* in the sense that they can be combined to form any other logic operation.

Logic operations are performed on a pixel by pixel basis between corresponding pixels of two or more images (except NOT, which operates on the pixels of a single image). Because the AND operation of two binary variables is 1 only when both variables are 1, the result at any location in a resulting AND image is 1 only if the corresponding pixels in the two input images are 1. Figure 9.3 shows various examples of logic operations involving images, where black indicates a binary 1 and white indicates a 0. (We use both conventions in this chapter, sometimes reversing the binary meaning of dark [black or gray] and light [white], depending on which is clearer in a given situation.) Other logic operations are easily constructed using the definitions in Table 9.1. For instance, the XOR (exclusive OR) operation yields a 1 when one or the other pixel (but *not* both) is 1, and it yields a 0 otherwise. This operation is unlike the OR operation, which is 1 when one or the other pixel is 1, or when both pixels are 1. Similarly, the NOT-AND operation selects the black pixels that simultaneously are in B, and not in A.

It is important to note that the logic operations just described have a one-to-one correspondence with the set operations discussed in Section 9.1.1, with the limitation that logic operations are restricted to binary variables, which is not the case in general for set operations. Thus, for example, the intersection operation in set theory reduces to the AND operation when the variables involved are binary. Terms such as *intersection* and *AND* (and even their notation) often are used interchangeably in the literature to denote general or binary set operations, with the meaning generally being clear from the context of the discussion.

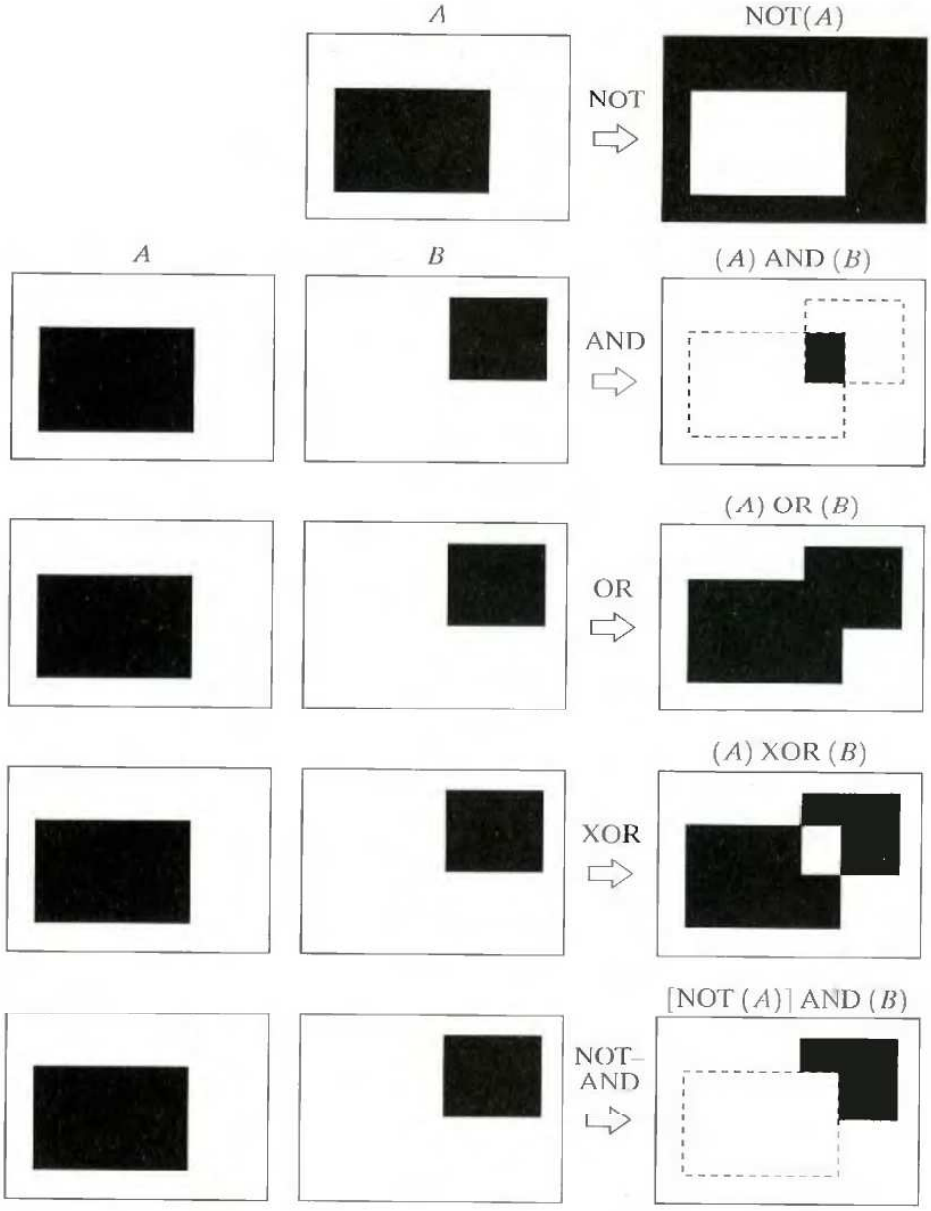


FIGURE 9.3 Some logic operations between binary images. Black represents binary 1s and white binary 0s in this example.

9.2 Dilation and Erosion

We begin the discussion of morphological operations by treating in some detail two operations: *dilation* and *erosion*. These operations are fundamental to morphological processing. In fact, many of the morphological algorithms discussed in this chapter are based on these two primitive operations.

9.2.1 Dilation

With A and B as sets in Z^2, the *dilation* of A by B, denoted $A \oplus B$, is defined as

$$A \oplus B = \{z \mid (\hat{B})_z \cap A \neq \varnothing\}. \tag{9.2-1}$$

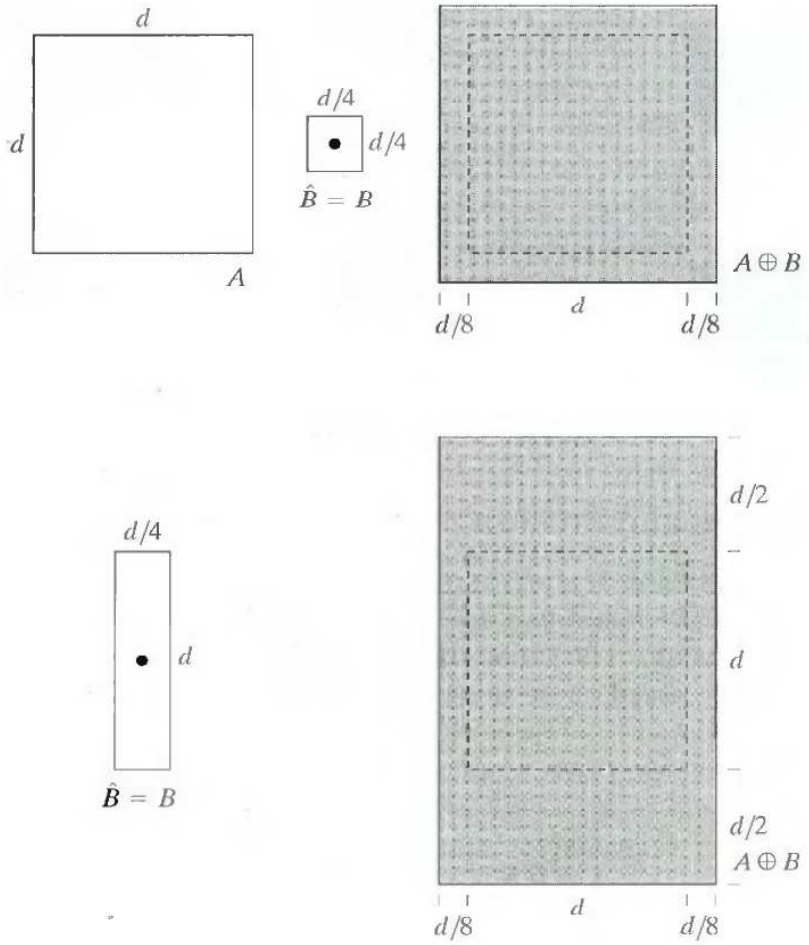
This equation is based on obtaining the reflection of B about its origin and shifting this reflection by z. The dilation of A by B then is the set of all *displacements*, z, such that $\hat{B}$ and A overlap by at least one element. Based on this interpretation, Eq. (9.2-1) may be rewritten as

$$A \oplus B = \{z | [(\hat{B})_z \cap A] \subseteq A\}. \tag{9.2-2}$$

Set B is commonly referred to as the *structuring element* in dilation, as well as in other morphological operations.

Equation (9.2-1) is not the only definition of dilation in the current literature on morphology (see Problems 9.10 and 9.11 for two different, yet equivalent, definitions). However, the preceding definition has a distinct advantage over other formulations in that it is more intuitive when the structuring element B is viewed as a convolution mask. Although dilation is based on set operations, whereas convolution is based on arithmetic operations, the basic process of "flipping" B about its origin and then successively displacing it so that it slides over set (image) A is analogous to the convolution process discussed in Sections 3.5 and 4.2.4.

Figure 9.4(a) shows a simple set, and Fig. 9.4(b) shows a structuring element and its reflection (the dark dot denotes the origin of the element). In this case the



a b c
d e

FIGURE 9.4 (a) Set A. (b) Square structuring element (dot is the center). (c) Dilation of A by B, shown shaded. (d) Elongated structuring element. (e) Dilation of A using this element.

structuring element and its reflection are equal because B is symmetric with respect to its origin. The dashed line in Fig. 9.4(c) shows the original set for reference, and the solid line shows the limit beyond which any further displacements of the origin of $\hat{B}$ by z would cause the intersection of $\hat{B}$ and A to be empty. Therefore, all points inside this boundary constitute the dilation of A by B. Figure 9.4(d) shows a structuring element designed to achieve more dilation vertically than horizontally. Figure 9.4(e) shows the dilation achieved with this element.

EXAMPLE 9.1: Use of morphological dilation for bridging gaps.

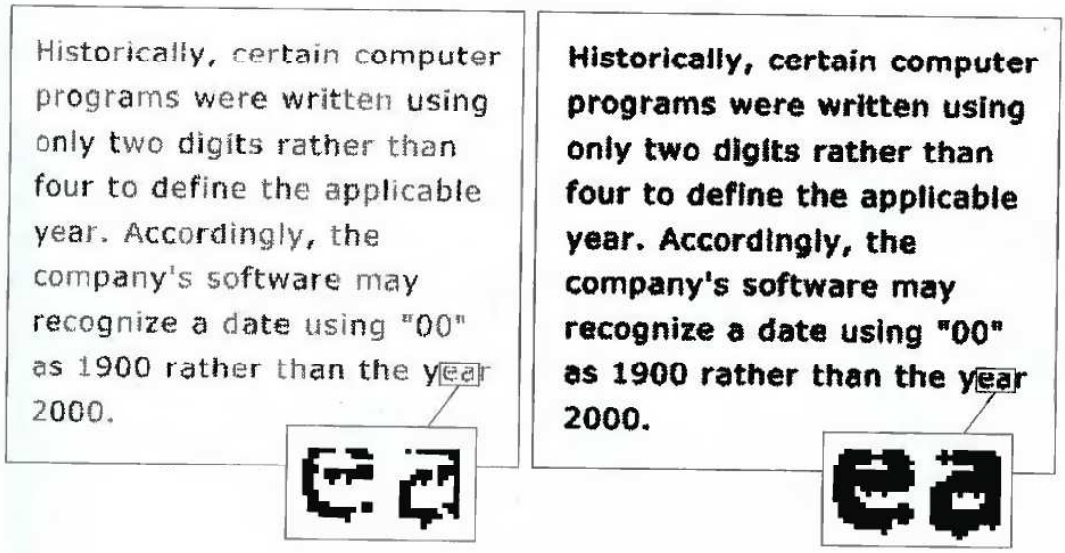
■ One of the simplest applications of dilation is for bridging gaps. Figure 9.5(a) shows the same image with broken characters that we studied in Fig. 4.19 in connection with lowpass filtering. The maximum length of the breaks is known to be two pixels. A simple structuring element that can be used for repairing the gaps is shown in Fig. 9.5(b). The result of dilating the original image with this structuring element is shown in Fig. 9.5(c). The gaps have been bridged. One immediate advantage of the morphological approach over the lowpass filtering method we used to bridge the gaps in Fig. 4.19 is that the morphological method resulted directly in a binary image. Lowpass filtering, on the other hand, started with a binary image and produced a gray-scale image, which would require a pass with a thresholding function to convert it back to binary form. ■

9.2.2 Erosion

For sets A and B in Z^2 the erosion of A by B, denoted $A \ominus B$, is defined as

$$A \ominus B = \{z \mid (B)_z \subseteq A\}. \tag{9.2-3}$$

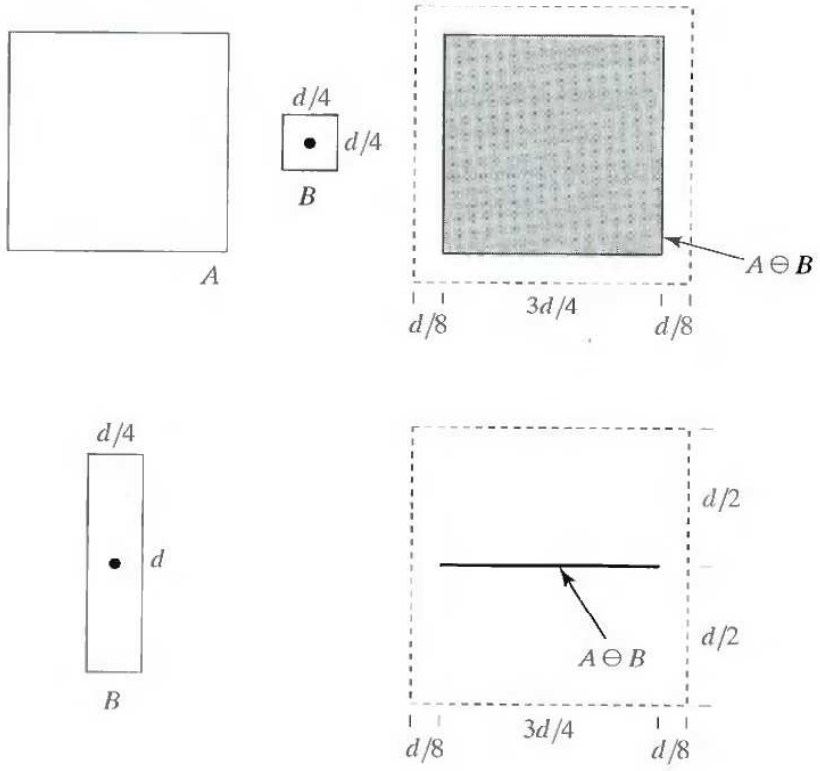
In words, this equation indicates that the erosion of A by B is the set of all points z such that B, translated by z, is contained in A. As in the case of dilation,



a c
b

FIGURE 9.5 (a) Sample text of poor resolution with broken characters (magnified view). (b) Structuring element. (c) Dilation of (a) by (b). Broken segments were joined.

0	1	0
1	1	1
0	1	0



a b c
d e

FIGURE 9.6 (a) Set A. (b) Square structuring element. (c) Erosion of A by B, shown shaded. (d) Elongated structuring element. (e) Erosion of A using this element.

Eq. (9.2-3) is not the only definition of erosion. (See Problems 9.12 and 9.13 for two different, yet equivalent, definitions.) However, Eq. (9.2-3) usually is favored in practical implementations of morphology for the same reasons stated earlier in connection with Eq. (9.2-1).

Figure 9.6 shows a process similar to that shown in Fig. 9.4. As before, set A is shown as a dashed line for reference in Fig. 9.6(c). The boundary of the shaded region shows the limit beyond which further displacement of the origin of B would cause this set to cease being completely contained in A. Thus, the locus of points within this boundary (i.e., the shaded region) constitutes the erosion of A by B. Figure 9.6(d) shows an elongated structuring element, and Fig. 9.6(e) shows the erosion of A by this element. Note that the original set was eroded down to a line.

Dilation and erosion are duals of each other with respect to set complementation and reflection. That is,

$$(A \ominus B)^c = A^c \oplus \hat{B}. \tag{9.2-4}$$

We proceed to prove this result formally in order to illustrate a typical approach for establishing the validity of morphological expressions. Starting with the definition of erosion, we have

$$(A \ominus B)^c = \{z \mid (B)_z \subseteq A\}^c.$$

If set $(B)_z$ is contained in set A, then $(B)_z \cap A^c = \varnothing$, in which case the preceding equation becomes

$$(A \ominus B)^c = \{z \mid (B)_z \cap A^c = \varnothing\}^c.$$

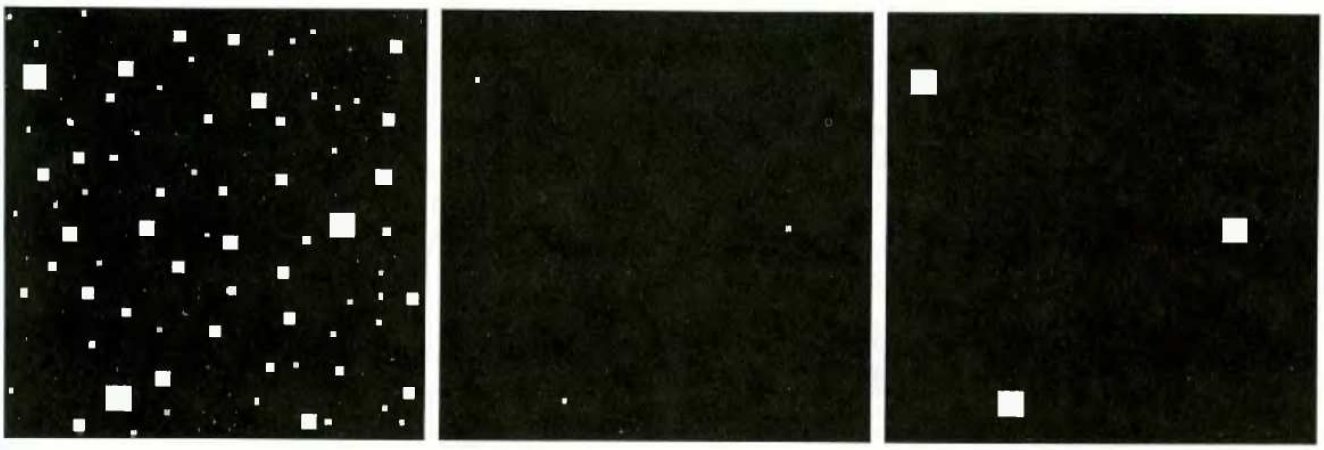
But the complement of the set of z's that satisfy $(B)_z \cap A^c = \varnothing$ is the set of z's such that $(B)_z \cap A^c \neq \varnothing$. Thus

$$\begin{aligned}(A \ominus B)^c &= \{z \mid (B)_z \cap A^c \neq \varnothing\} \\ &= A^c \oplus \hat{B}\end{aligned}$$

where the last step follows from Eq. (9.2-1). This concludes the proof.

EXAMPLE 9.2: Use of morphological erosion for removing image components.

■ One of the simplest uses of erosion is for eliminating irrelevant detail (in terms of size) from a binary image. Figure 9.7(a) shows a binary image composed of squares of sizes 1, 3, 5, 7, 9, and 15 pixels on the side. Suppose that we want to eliminate all the squares except the largest ones. We can do this by eroding the image with a structuring element of a size somewhat smaller than the objects we wish to keep. In this example we chose a structuring element of size 13×13 pixels. The result of eroding the original image with this structuring element is shown in Fig. 9.7(b). Only portions of the largest squares remain. As shown in Fig. 9.7(c), we can restore these three squares to their original 15×15 size by dilating them with the same structuring element we used for erosion (dilation does not fully restore eroded objects in general; see Problem 9.18). Note in all three images in this example that objects are represented by white pixels, rather than by black pixels as in the previous example. As noted earlier, both representations are used in practice. Unless it is stated otherwise, it is generally understood that the "active" elements of structuring elements assume the same binary values as the objects of interest. The concepts presented in this example are the basis for morphological filtering, as discussed in the following section. ■



a b c

FIGURE 9.7 (a) Image of squares of size 1, 3, 5, 7, 9, and 15 pixels on the side. (b) Erosion of (a) with a square structuring element of 1's, 13 pixels on the side. (c) Dilation of (b) with the same structuring element.

9.3 Opening and Closing

As we have seen, dilation expands an image and erosion shrinks it. In this section we discuss two other important morphological operations: opening and closing. *Opening* generally smoothes the contour of an object, breaks narrow isthmuses, and eliminates thin protrusions. *Closing* also tends to smooth sections of contours but, as opposed to opening, it generally fuses narrow breaks and long thin gulfs, eliminates small holes, and fills gaps in the contour.

The opening of set A by structuring element B, denoted $A \circ B$, is defined as

$$A \circ B = (A \ominus B) \oplus B. \tag{9.3-1}$$

Thus, the opening A by B is the erosion of A by B, followed by a dilation of the result by B.

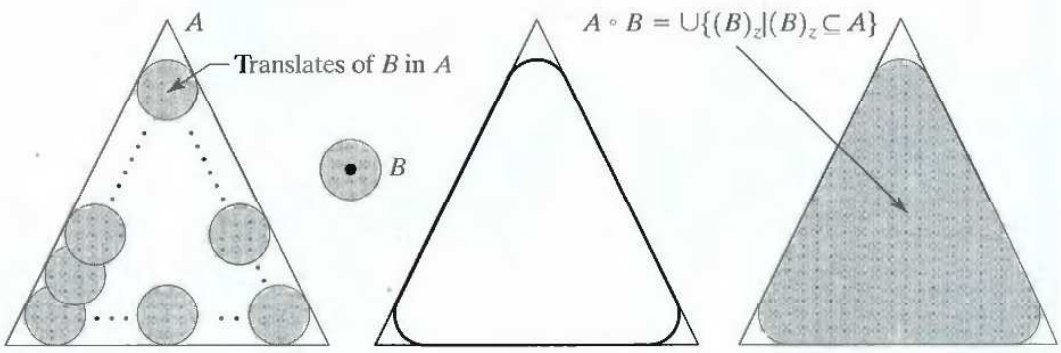
Similarly, the closing of set A by structuring element B, denoted $A \bullet B$, is defined as

$$A \bullet B = (A \oplus B) \ominus B, \tag{9.3-2}$$

which, in words, says that the closing of A by B is simply the dilation of A by B, followed by the erosion of the result by B.

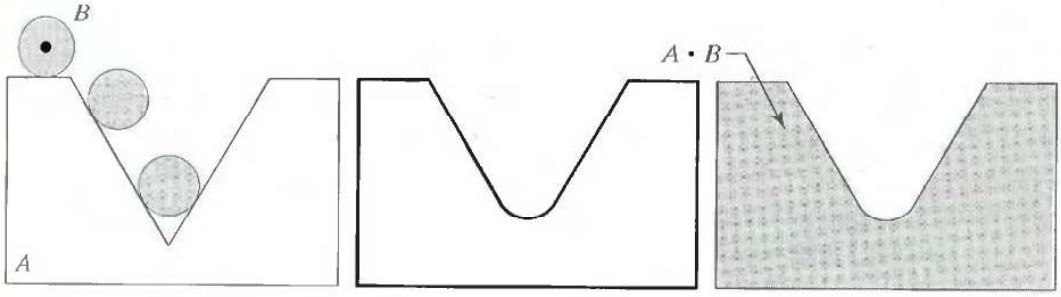
The opening operation has a simple geometric interpretation (Fig. 9.8). Suppose that we view the structuring element B as a (flat) "rolling ball." The *boundary* of $A \circ B$ is then established by the points in B that reach the *farthest* into the boundary of A as B is rolled around the *inside* of this boundary. This geometric *fitting* property of the opening operation leads to a set-theoretic formulation, which states that the opening of A by B is obtained by taking the union of all translates of B that fit into A. That is, opening can be expressed as a fitting process such that

$$A \circ B = \cup\{(B)_z \mid (B)_z \subseteq A\} \tag{9.3-3}$$



a b c d

FIGURE 9.8 (a) Structuring element B "rolling" along the inner boundary of A (the dot indicates the origin of B). (c) The heavy line is the outer boundary of the opening. (d) Complete opening (shaded).



a b c

FIGURE 9.9 (a) Structuring element B "rolling" on the outer boundary of set A. (b) Heavy line is the outer boundary of the closing. (c) Complete closing (shaded).

where $\cup\{\cdot\}$ denotes the union of all the sets inside the braces.

Closing has a similar geometric interpretation, except that now we roll B on the outside of the boundary (Fig. 9.9). It will be shown shortly that opening and closing are duals of each other, so having to roll the ball on the outside is not unexpected. Geometrically, a point w is an element of $A \bullet B$ if and only if $(B)_z \cap A \neq \varnothing$ for any translate of $(B)_z$ that contains w. Figure 9.9 illustrates the basic geometrical properties of closing.

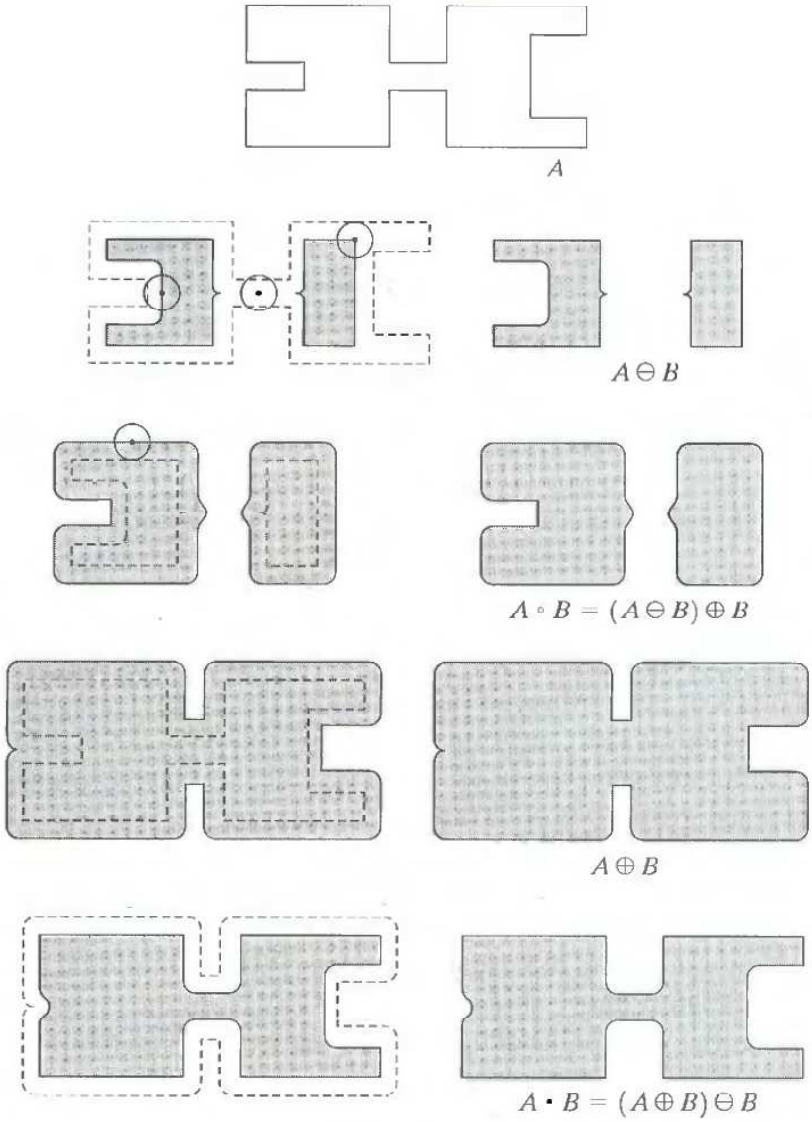
EXAMPLE 9.3: A simple illustration of morphological opening and closing.

■ Figure 9.10 further illustrates the opening and closing operations. Figure 9.10(a) shows a set A, and Fig. 9.10(b) shows various positions of a disk structuring element during the erosion process. When completed, this process resulted in the disjoint figure shown in Fig. 9.10(c). Note the elimination of the bridge between the two main sections. Its width was thin in relation to the diameter of the structuring element; that is, the structuring element could not be completely contained in this part of the set, thus violating the conditions of Eq. (9.2-3). The same also was true of the two rightmost members of the object. Protruding elements where the disk did not fit were eliminated. Figure 9.10(d) shows the process of dilating the eroded set, and Fig. 9.10(e) shows the final result of opening. Note that outward pointing corners were rounded, whereas inward pointing corners were not affected.

Similarly, Figs. 9.10(f) through (i) show the results of closing A with the same structuring element. We note that the inward pointing corners were rounded, whereas the outward pointing corners remained unchanged. The leftmost intrusion on the boundary of A was reduced in size significantly, because the disk did not fit there. Note also the smoothing that resulted in parts of the object from both opening and closing the set A with a circular structuring element. ■

As in the case of dilation and erosion, opening and closing are duals of each other with respect to set complementation and reflection. That is,

$$(A \bullet B)^c = (A^c \circ \hat{B}). \tag{9.3-4}$$



a
b c
d e
f g
h i

FIGURE 9.10 Morphological opening and closing. The structuring element is the small circle shown in various positions in (b). The dark dot is the center of the structuring element.

We leave the proof of this result as an exercise (Problem 9.14).

The opening operation satisfies the following properties:

(i) $A \circ B$ is a subset (subimage) of A.
(ii) If C is a subset of D, then $C \circ B$ is a subset of $D \circ B$.
(iii) $(A \circ B) \circ B = A \circ B$.

Similarly, the closing operation satisfies the following properties:

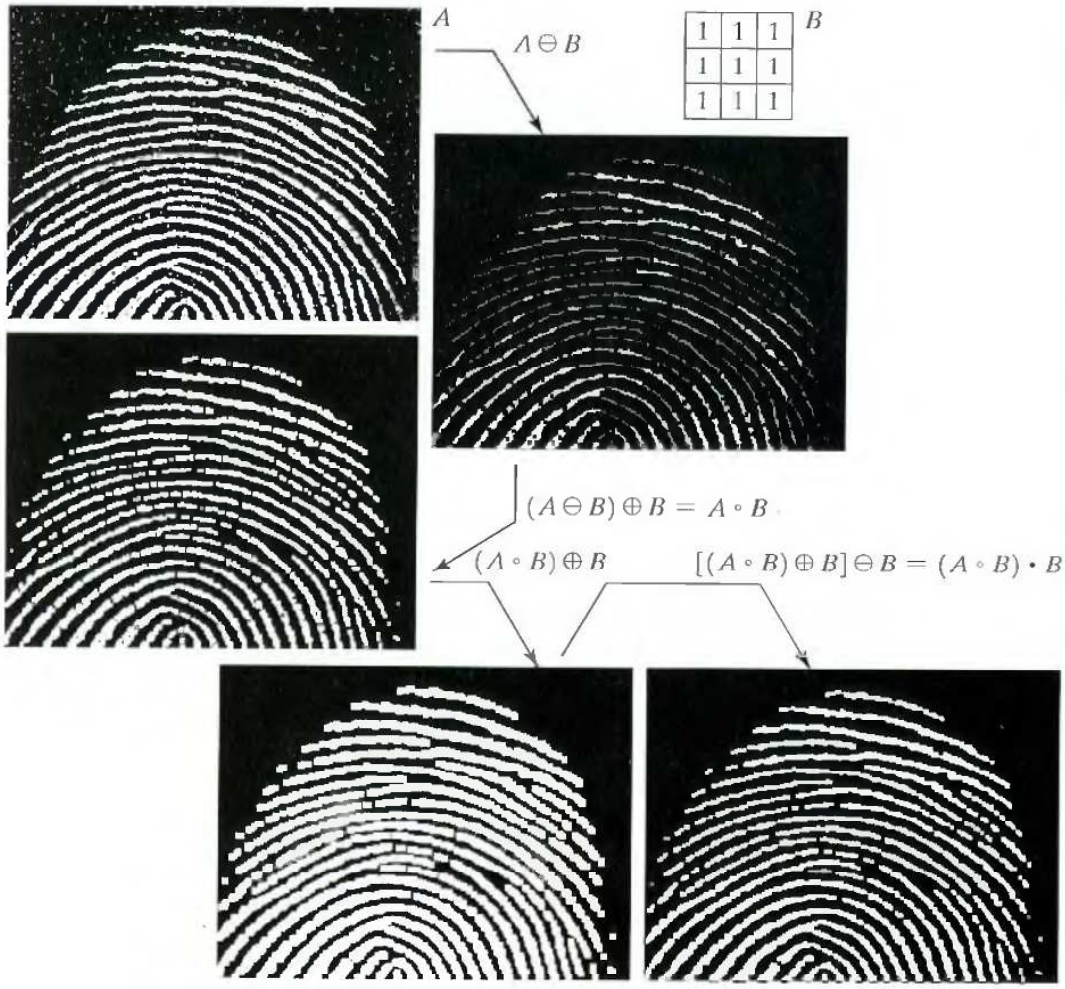
(i) A is a subset (subimage) of $A \bullet B$.
(ii) If C is a subset of D, then $C \bullet B$ is a subset of $D \bullet B$.
(iii) $(A \bullet B) \bullet B = A \bullet B$.

Note from condition (*iii*) in both cases that multiple openings or closings of a set have no effect after the operator has been applied once.

EXAMPLE 9.4: Use of opening and closing for morphological filtering.

Morphological operations can be used to construct filters similar in concept to the spatial filters discussed in Chapter 3. The binary image shown in Fig. 9.11(a) shows a section of a fingerprint corrupted by noise. Here the noise manifests itself as light elements on a dark background and as dark elements on the light components of the fingerprint. The objective is to eliminate the noise and its effects on the print while distorting it as little as possible. A morphological filter consisting of opening followed by closing can be used to accomplish this objective.

The structuring element used is shown in Fig. 9.11(b). The rest of Fig. 9.11 shows a step-by-step sequence of the filtering operation. Figure 9.11(c) shows the result of eroding A with the structuring element. The background noise was completely eliminated in the erosion stage of opening because in this case all noise components are physically smaller than the structuring element. The size of the noise elements (dark spots) contained within the fingerprint actually increased in size. The reason is that these elements actually are inner boundaries that should increase in size as the object is eroded. This enlargement is countered by performing dilation on Fig. 9.11(c). Figure 9.11(d) shows the result. The noise components contained in the fingerprint were reduced in size or deleted completely.



a b
c d
e f

FIGURE 9.11 (a) Noisy image. (b) Structuring element. (c) Eroded image. (d) Opening of A. (e) Dilation of the opening. (f) Closing of the opening. (Original image for this example courtesy of the National Institute of Standards and Technology.)

The two operations just described constitute the opening of A by B. We note in Fig. 9.11(d) that the net effect of opening was to eliminate virtually all noise components in both the background and the fingerprint itself. However, new gaps between the fingerprint ridges were created. To counter this undesirable effect, we perform a dilation on the opening, as shown in Fig. 9.11(e). Most of the breaks were restored, but the ridges were thickened, a condition that can be remedied by erosion. The result, shown in Fig. 9.11(f), constitutes the closing of the opening in Fig. 9.11(d). This final result is remarkably clean of noise specks, but it has the disadvantage that some of the print ridges were not fully repaired, and thus contain breaks. This is not totally unexpected, because no conditions were built into the procedure for maintaining connectivity (we discuss this issue again in Example 9.8 and give ways to address it in Section 11.1.5). ■

9.4 The Hit-or-Miss Transformation

The morphological hit-or-miss transform is a basic tool for shape detection. We introduce this concept with the aid of Fig. 9.12, which shows a set A consisting of three shapes (subsets), denoted X, Y, and Z. The shading in Figs. 9.12(a) through (c) indicates the original sets, whereas the shading in Figs. 9.12(d) and (e) indicates the result of morphological operations. The objective is to find the location of one of the shapes, say, X.

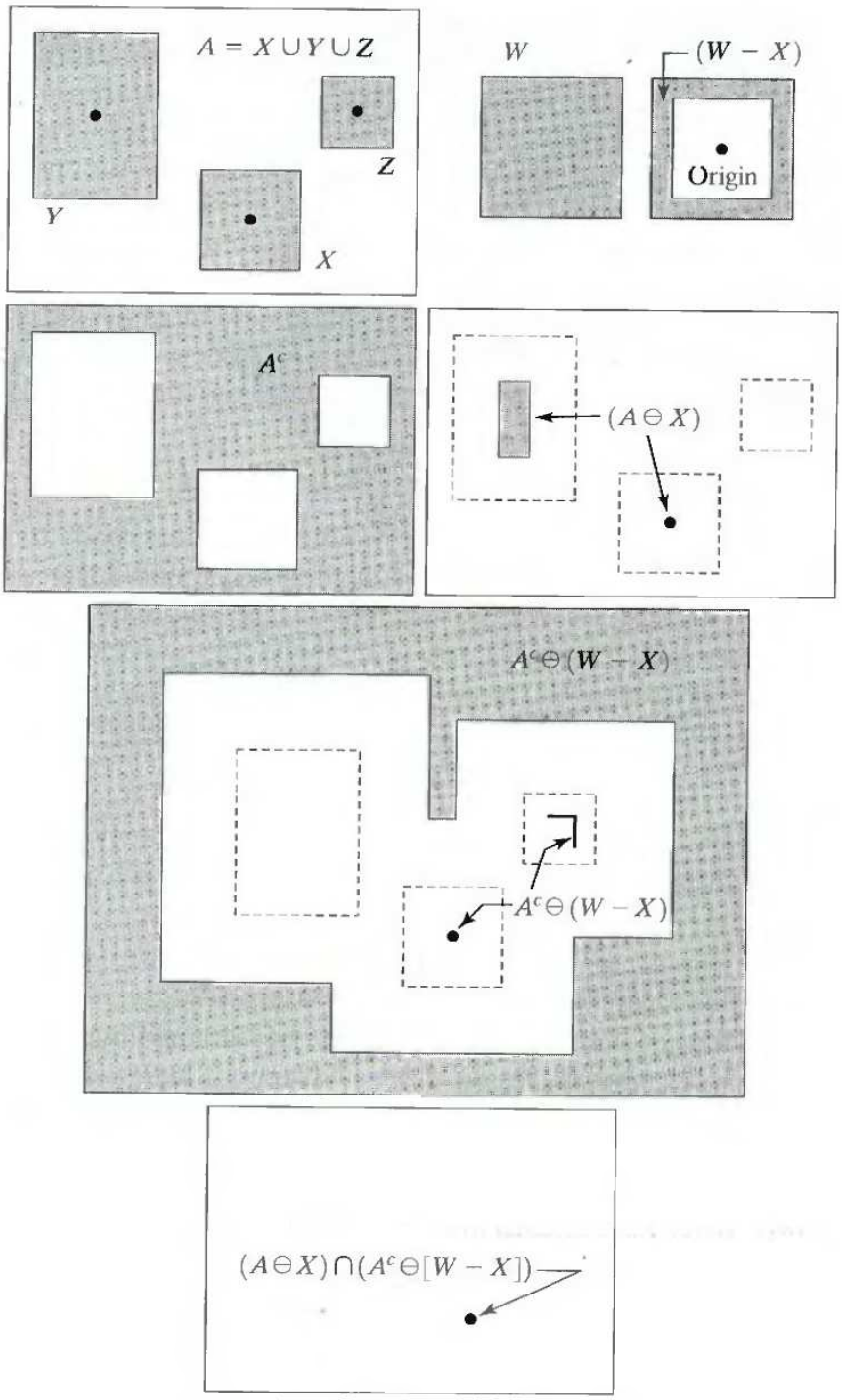
Let the origin of each shape be located at its center of gravity. Let X be enclosed by a small window, W. The *local background* of X with respect to W is defined as the set difference $(W - X)$, as shown in Fig. 9.12(b). Figure 9.12(c) shows the complement of A, which is needed later. Figure 9.12(d) shows the erosion of A by X (the dashed lines are included for reference). Recall that the erosion of A by X is the set of locations of the *origin* of X, such that X is completely contained in A. Interpreted another way, $A \ominus X$ may be viewed geometrically as the set of all locations of the origin of X at which X found a match (hit) in A. Keep in mind that in Fig. 9.12 A consists only of the three *disjoint* sets X, Y, and Z.

Figure 9.12(e) shows the erosion of the complement of A by the local background set $(W - X)$. The outer shaded region in Fig. 9.12(e) is part of the erosion. We note from Figs. 9.12(d) and (e) that the set of locations for which X *exactly* fits inside A is the *intersection* of the erosion of A by X and the erosion of A^c by $(W - X)$ as shown in Fig. 9.12(f). This intersection is precisely the location sought. In other words, if B denotes the set composed of X and its background, the match (or set of matches) of B in A, denoted $A \circledast B$, is

$$A \circledast B = (A \ominus X) \cap \left[A^c \ominus (W - X)\right]. \tag{9.4-1}$$

We can generalize the notation somewhat by letting $B = (B_1, B_2)$, where B_1 is the set formed from elements of B associated with an object and B_2 is the set of elements of B associated with the corresponding background. From the preceding discussion, $B_1 = X$ and $B_2 = (W - X)$. With this notation, Eq. (9.4-1) becomes

$$A \circledast B = (A \ominus B_1) \cap (A^c \ominus B_2). \tag{9.4-2}$$



a b
c d
e
f

FIGURE 9.12 (a) Set A. (b) A window, W, and the local background of X with respect to W, $(W - X)$. (c) Complement of A. (d) Erosion of A by X. (e) Erosion of A^c by $(W - X)$. (f) Intersection of (d) and (e), showing the location of the origin of X, as desired.

Thus, set $A \circledast B$ contains all the (origin) points at which, simultaneously, B_1 found a match ("hit") in A and B_2 found a match in A^c. By using the definition of set differences given in Eq. (9.1-8) and the dual relationship between erosion and dilation given in Eq. (9.2-4), we can write Eq. (9.4-2) as

$$A \circledast B = (A \ominus B_1) - (A \oplus \hat{B}_2). \tag{9.4-3}$$

However, Eq. (9.4-2) is considerably more intuitive. We refer to any of the preceding three equations as the *morphological hit-or-miss transform*.

The reason for using a structuring element B_1 associated with objects and an element B_2 associated with the background is based on an assumed definition that two or more objects are distinct only if they form disjoint (disconnected) sets. This is guaranteed by requiring that each object have at least a one-pixel-thick background around it. In some applications, we may be interested in detecting certain patterns (combinations) of 1's and 0's within a set, in which case a background is not required. In such an instance, the hit-or-miss transform reduces to simple erosion. As indicated previously, erosion is still a set of matches, but without the additional requirement of a background match for detecting individual objects. This simplified pattern detection scheme is used in some of the algorithms developed in the following section.

9.5 Some Basic Morphological Algorithms

With the preceding discussion as background, we are now ready to consider some practical uses of morphology. When dealing with binary images, the principal application of morphology is extracting image components that are useful in the representation and description of shape. In particular, we consider morphological algorithms for extracting boundaries, connected components, the convex hull, and the skeleton of a region. We also develop several methods (for region filling, thinning, thickening, and pruning) that are used frequently in conjunction with these algorithms as pre- or postprocessing steps. We make extensive use in this section of "mini-images," designed to clarify the mechanics of each morphological process as we introduce it. The images are binary, with 1's shown shaded and 0's shown in white.

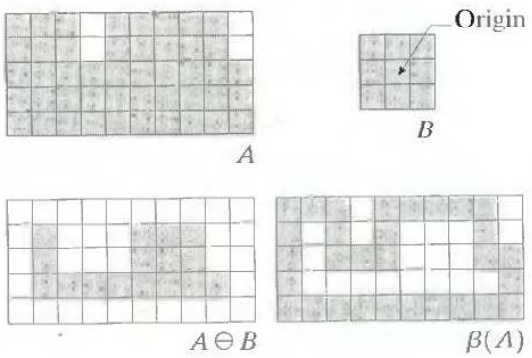
9.5.1 Boundary Extraction

The boundary of a set A, denoted by $\beta(A)$, can be obtained by first eroding A by B and then performing the set difference between A and its erosion. That is,

$$\beta(A) = A - (A \ominus B) \tag{9.5-1}$$

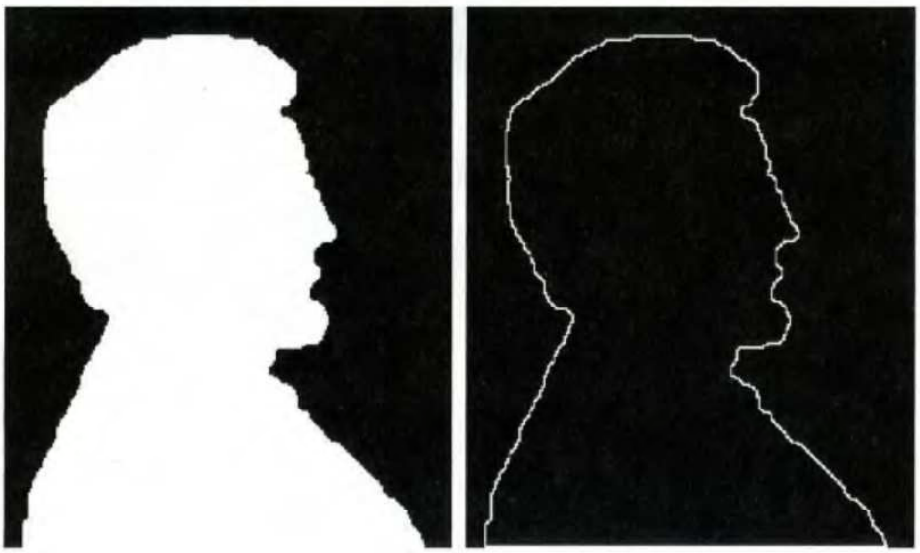
where B is a suitable structuring element.

Figure 9.13 illustrates the mechanics of boundary extraction. It shows a simple binary object, a structuring element B, and the result of using Eq. (9.5-1).



a b
c d

FIGURE 9.13 (a) Set A. (b) Structuring element B. (c) A eroded by B. (d) Boundary, given by the set difference between A and its erosion.



a b

FIGURE 9.14 (a) A simple binary image, with 1's represented in white. (b) Result of using Eq. (9.5-1) with the structuring element in Fig. 9.13(b).

Although the structuring element shown in Fig. 9.13(b) is among the most frequently used, it is by no means unique. For example, using a 5×5 structuring element of 1's would result in a boundary between 2 and 3 pixels thick. Note that when the origin of B is on the edges of the set, part of the structuring element may be outside the image. The normal treatment of this condition is to assume that values outside the borders of the image are 0.

EXAMPLE 9.5: Boundary extraction by morphological processing.

■ Figure 9.14 further illustrates the use of Eq. (9.5-1) with the structuring element of Fig. 9.13(b). In this example, binary 1's are shown in white and 0's in black, so the elements of the structuring element, which are 1's, also are treated as white. Because of the structuring element used, the boundary shown in Fig. 9.14(b) is one pixel thick. ■

9.5.2 Region Filling

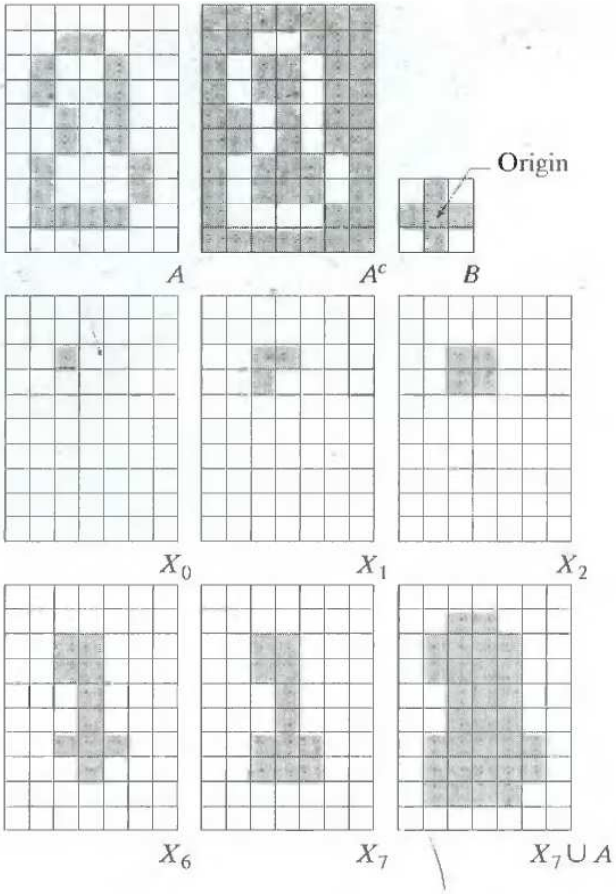
Next we develop a simple algorithm for region filling based on set dilations, complementation, and intersections. In Fig. 9.15 A denotes a set containing a subset whose elements are 8-connected boundary points of a region. Beginning with a point p inside the boundary, the objective is to fill the entire region with 1's.

If we adopt the convention that all nonboundary (background) points are labeled 0, then we assign a value of 1 to p to begin. The following procedure then fills the region with 1's:

$$X_k = (X_{k-1} \oplus B) \cap A^c \quad k = 1, 2, 3, \ldots \tag{9.5-2}$$

where $X_0 = p$, and B is the symmetric structuring element shown in Fig. 9.15(c). The algorithm terminates at iteration step k if $X_k = X_{k-1}$. The set union of X_k and A contains the filled set and its boundary.

The dilation process of Eq. (9.5-2) would fill the entire area if left unchecked. However, the intersection at each step with A^c limits the result to inside the region of interest. This is our first example of how a morphological process can be conditioned to meet a desired property. In the current application, it is



a b c
d e f
g h i

FIGURE 9.15 Region filling. (a) Set A. (b) Complement of A. (c) Structuring element B. (d) Initial point inside the boundary. (e)–(h) Various steps of Eq. (9.5-2). (i) Final result [union of (a) and (h)].

appropriately called *conditional dilation*. The rest of Fig. 9.15 illustrates further the mechanics of Eq. (9.5-2). Although this example has only one subset, the concept clearly applies to any finite number of such subsets, assuming that a point inside each boundary is given.

EXAMPLE 9.6: Morphological region filling.

■ Figure 9.16(a) shows an image composed of white circles with black inner spots. An image such as this might result from thresholding into two levels a scene containing polished spheres (e.g., ball bearings). The dark spots inside the spheres are the result of reflections. The objective is to eliminate the reflections by region filling. Figure 9.16(a) shows one point selected inside one of the spheres, and Fig. 9.16(b) shows the result of filling that component. Finally, Fig. 9.16(c) shows the result of filling all the spheres. Because it must be known whether black points are background points or sphere inner points, fully automating this procedure requires that additional "intelligence" be built into the algorithm (see Problem 9.23). ■

9.5.3 Extraction of Connected Components

The concepts of connectivity and connected components were introduced in Section 2.5.2. In practice, extraction of connected components in a binary image is central to many automated image analysis applications. Let Y represent a connected component contained in a set A and assume that a point p of Y is

a b c

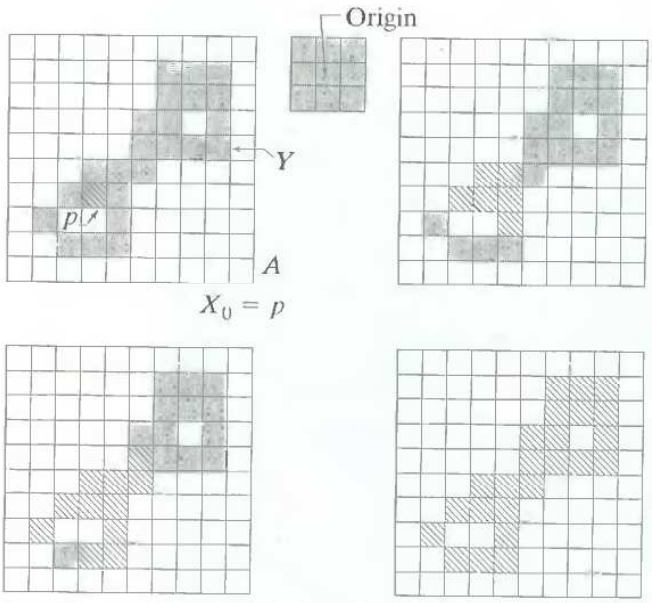
FIGURE 9.16 (a) Binary image (the white dot inside one of the regions is the starting point for the region-filling algorithm). (b) Result of filling that region (c) Result of filling all regions.

known. Then the following iterative expression yields all the elements of Y:

$$X_k = (X_{k-1} \oplus B) \cap A \quad k = 1, 2, 3, \ldots \tag{9.5-3}$$

where $X_0 = p$, and B is a suitable structuring element, as shown in Fig. 9.17. If $X_k = X_{k-1}$, the algorithm has converged and we let $Y = X_k$.

Equation (9.5-3) is similar in form to Eq. (9.5-2). The only difference is the use of A instead of its complement. This difference arises because all the elements sought (that is, the elements of the connected component) are labeled 1.



a b c
d e

FIGURE 9.17 (a) Set A showing initial point p (all shaded points are valued 1, but are shown different from p to indicate that they have not yet been found by the algorithm). (b) Structuring element. (c) Result of first iterative step. (d) Result of second step. (e) Final result.

The intersection with A at each iterative step eliminates dilations centered on elements labeled 0. Figure 9.17 illustrates the mechanics of Eq. (9.5-3). Note that the shape of the structuring element assumes 8-connectivity between pixels. As in the region-filling algorithm, the results just discussed are applicable to any finite number of sets of connected components contained in A, assuming that a point is known in each connected component.

EXAMPLE 9.7: Using connected components to detect foreign objects in packaged food.

■ Connected components are used frequently for automated inspection. Figure 9.18(a) shows an X-ray image of a chicken breast that contains bone fragments. It is of considerable interest to be able to detect such foreign objects in processed food before packaging and/or shipping. In this particular case, the density of the bones is such that their nominal gray-level values are different from the background. This make extraction of the bones from the background a simple matter by using a single threshold (thresholding was introduced in Section 3.1 and is discussed in considerable more detail in Section 10.3). The result is the binary image shown in Fig. 9.18(b).

The most significant feature in this figure is the fact that the points that remain are clustered into objects (bones), rather than being isolated, irrelevant points. We can make sure that only objects of "significant" size remain by erod-

a
b
c d

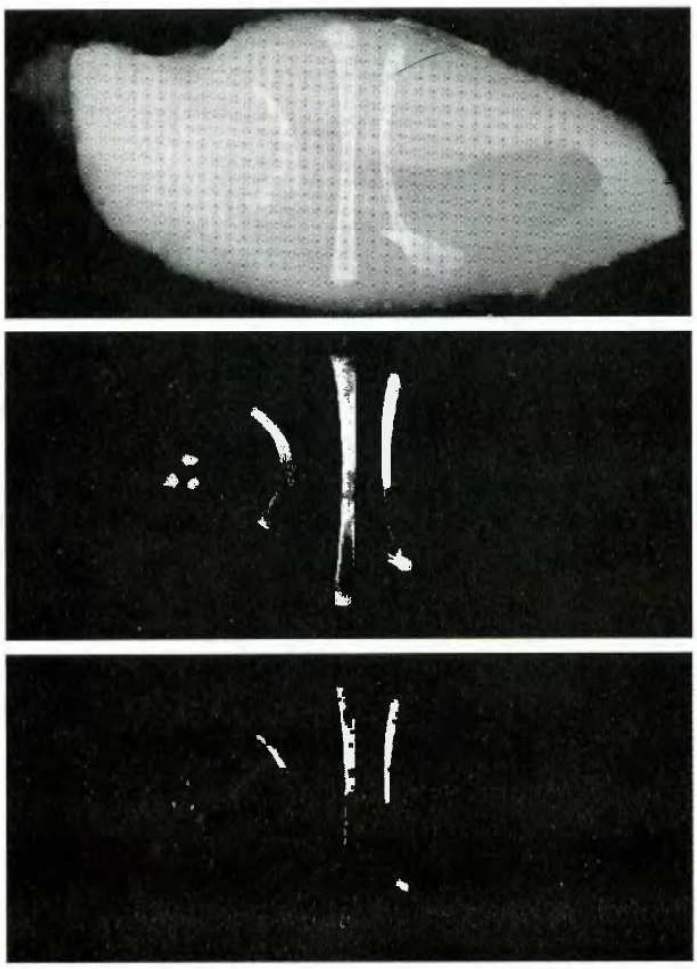


FIGURE 9.18 (a) X-ray image of chicken filet with bone fragments. (b) Thresholded image. (c) Image eroded with a 5×5 structuring element of 1's. (d) Number of pixels in the connected components of (c). (Image courtesy of NTB Elektronische Geraete GmbH, Diepholz, Germany, www.ntbxray.com.)

Connected component	No. of pixels in connected comp
01	11
02	9
03	9
04	39
05	133
06	1
07	1
08	743
09	7
10	11
11	11
12	9
13	9
14	674
15	85

ing the thresholded image. In this example, we define as significant any object that remains after erosion with a 5×5 structuring element of 1's. The result of erosion is shown in Fig. 9.18(c). The next step is to analyze the size of the objects that remain. We label (identify) these objects by extracting the connected components in the image. The table in Fig. 9.18(d) lists the results of the extraction. There are a total of 15 connected components, with four of them being dominant in size. This is enough to determine that significant foreign objects are contained in the original image. If desired, further characterization (such as shape) is possible using the techniques discussed in Chapter 11. ■

9.5.4 Convex Hull

A set A is said to be *convex* if the straight line segment joining any two points in A lies entirely within A. The *convex hull* H of an arbitrary set S is the smallest convex set containing S. The set difference $H - S$ is called the *convex deficiency* of S. As discussed in more detail in Sections 11.1.4 and 11.3.2, the convex hull and convex deficiency are useful for object description. Here, we present a simple morphological algorithm for obtaining the convex hull, $C(A)$, of a set A.

Let B^i, $i = 1, 2, 3, 4$, represent the four structuring elements in Fig. 9.19(a). The procedure consists of implementing the equation:

$$X_k^i = (X_{k-1} \circledast B^i) \cup A \quad i = 1, 2, 3, 4 \quad \text{and} \quad k = 1, 2, 3, \ldots \tag{9.5-4}$$

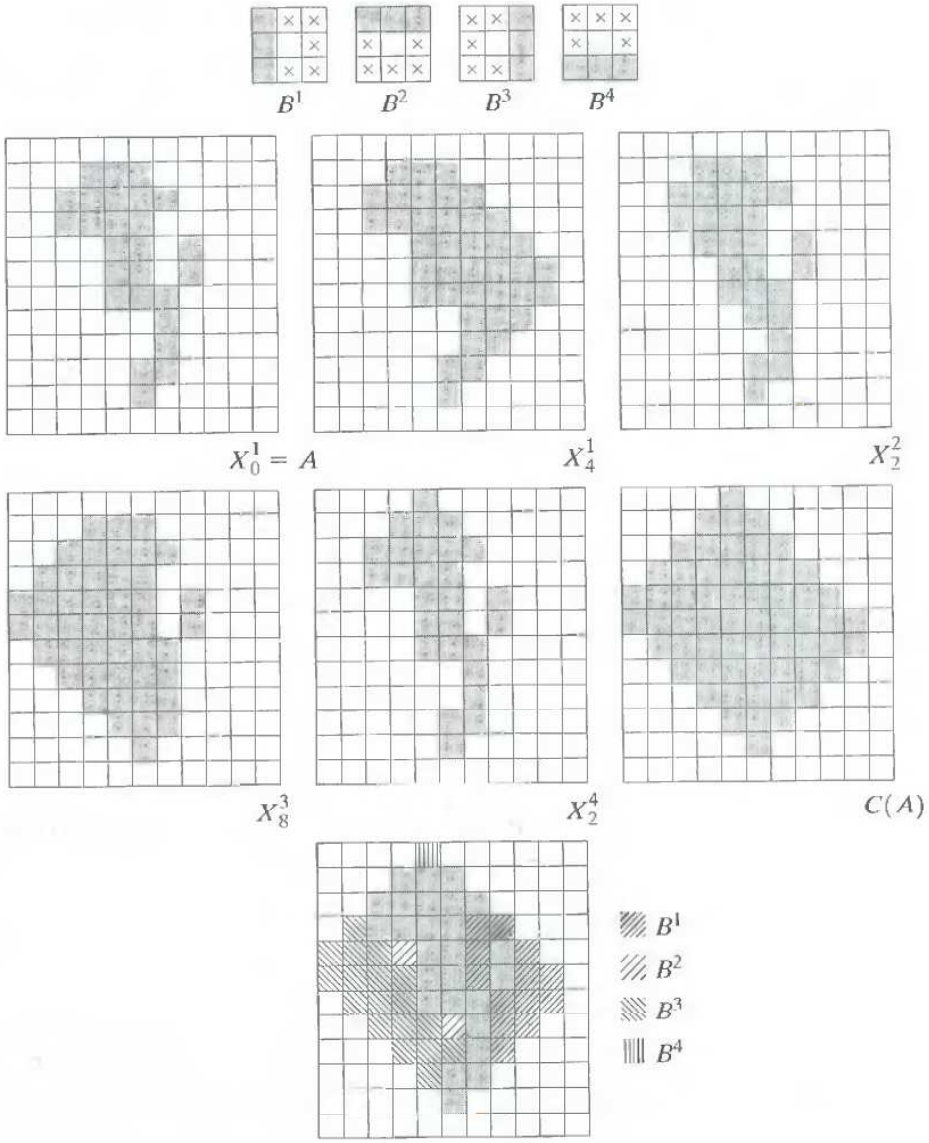
with $X_0^i = A$. Now let $D^i = X_{\text{conv}}^i$, where the subscript "conv" indicates convergence in the sense that $X_k^i = X_{k-1}^i$. Then the convex hull of A is

$$C(A) = \bigcup_{i=1}^{4} D^i. \tag{9.5-5}$$

In other words, the procedure consists of iteratively applying the hit-or-miss transform to A with B^1; when no further changes occur, we perform the union with A and call the result D^1. The procedure is repeated with B^2 (applied to A) until no further changes occur, and so on. The union of the four resulting D's constitutes the convex hull of A. Note that we are using the simplified implementation of the hit-or-miss transform in which no background match is required, as discussed at the end of Section 9.4.

Figure 9.19 illustrates the procedure given in Eqs. (9.5-4) and (9.5-5). Figure 9.19(a) shows the structuring elements used to extract the convex hull. The origin of each element is at its center. The $\times$ entries indicate "don't care" conditions. This means that a structuring element is said to have found a match in A if the 3-by-3 region of A under the structuring element mask at that location matches the pattern of the mask. For a particular mask, a pattern match occurs when the center of the 3-by-3 region in A is 0, and the three pixels under the shaded mask elements are 1. The values of the other pixels in the 3-by-3 region do not matter. Also, with respect to the notation in Fig. 9.19(a), B^i is a clockwise rotation of B^{i-1} by 90°.

Figure 9.19(b) shows a set A for which the convex hull is sought. Starting with $X_0^1 = A$ resulted in the set shown in Fig. 9.19(c) after four iterations of Eq. (9.5-4). Then, letting $X_0^2 = A$ and again using Eq. (9.5-4) resulted in the set



a
b c d
e f g
h

FIGURE 9.19 (a) Structuring elements. (b) Set A. (c)–(f) Results of convergence with the structuring elements shown in (a). (g) Convex hull. (h) Convex hull showing the contribution of each structuring element.

shown in Fig. 9.19(d) (convergence was achieved in only two steps in this case). The next two results were obtained in the same way. Finally, forming the union of the sets in Figs. 9.19(c), (d), (e), and (f) resulted in the convex hull shown in Fig. 9.19(g). The contribution of each structuring element is highlighted in the composite set shown in Fig. 9.19(h).

One obvious shortcoming of the procedure just outlined is that the convex hull can grow beyond the minimum dimensions required to guarantee convexity. One simple approach to reduce this effect is to limit growth so that it does not extend past the vertical and horizontal dimensions of the original set of points. Imposing this limitation on the example in Fig. 9.19 resulted in the image shown in Fig. 9.20. Boundaries of greater complexity can be used to limit growth even further in images with more detail. For example, we could use the maximum dimensions of the original set of points along the vertical, horizontal, and diagonal directions. The price paid for refinements such as this is additional complexity (and increased computational requirements) of the algorithm.

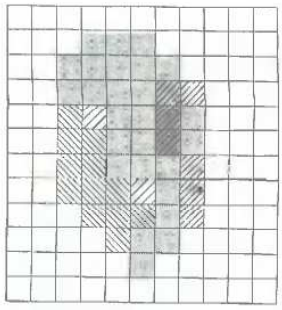


FIGURE 9.20 Result of limiting growth of convex hull algorithm to the maximum dimensions of the original set of points along the vertical and horizontal directions.

9.5.5 Thinning

The thinning of a set A by a structuring element B, denoted $A \otimes B$, can be defined in terms of the hit-or-miss transform:

$$\begin{aligned} A \otimes B &= A - (A \circledast B) \\ &= A \cap (A \circledast B)^c. \end{aligned} \tag{9.5-6}$$

As in the previous section, we are interested only in pattern matching with the structuring elements, so no background operation is required in the hit-or-miss transform. A more useful expression for thinning A symmetrically is based on a *sequence* of structuring elements:

$$\{B\} = \{B^1, B^2, B^3, \ldots, B^n\} \tag{9.5-7}$$

where B^i is a rotated version of B^{i-1}. Using this concept, we now define thinning by a sequence of structuring elements as

$$A \otimes \{B\} = ((\ldots((A \otimes B^1) \otimes B^2) \ldots) \otimes B^n). \tag{9.5-8}$$

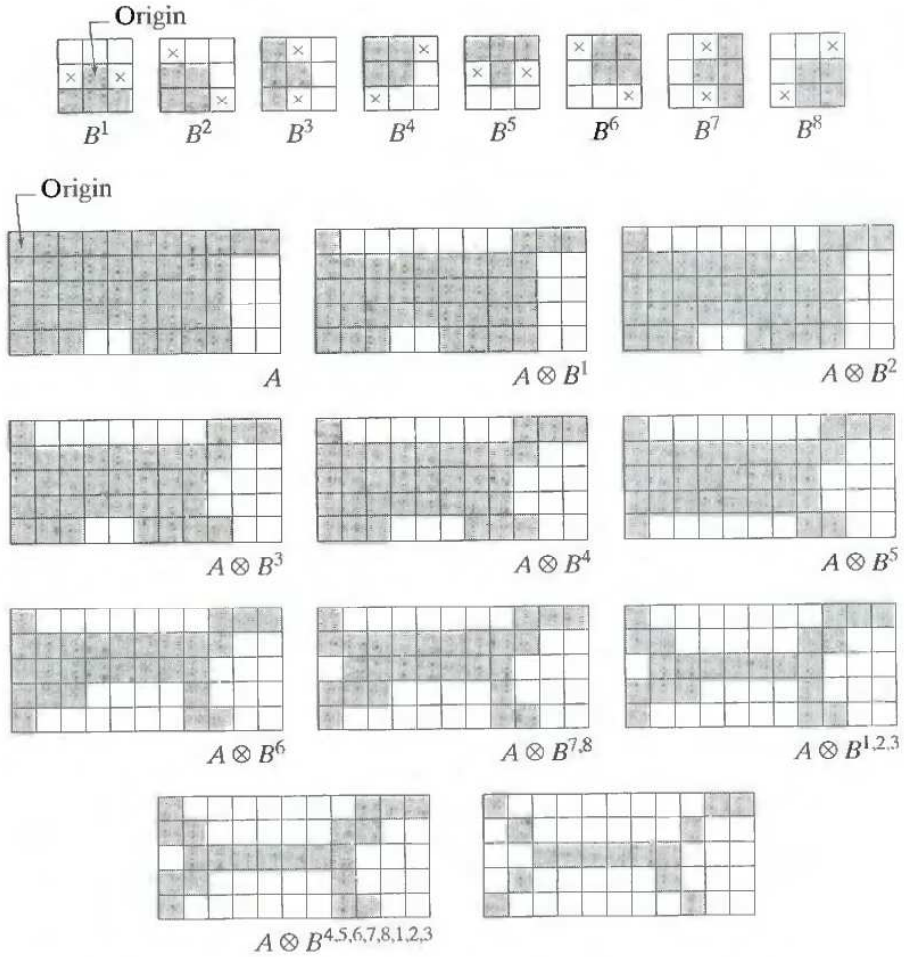
The process is to thin A by *one pass* with B^1, then thin the result with one pass of B^2, and so on, until A is thinned with one pass of B^n. The entire process is repeated until no further changes occur. Each individual thinning pass is performed using Eq. (9.5-6).

Figure 9.21(a) shows a set of structuring elements commonly used for thinning, and Fig. 9.21(b) shows a set A to be thinned by using the procedure just discussed. Figure 9.21(c) shows the result of thinning with one raster pass of A with B^1, and Figs. 9.21(d) through (k) show the results of passes with the other structuring elements. Convergence was achieved after the second pass of B^4. Figure 9.21(k) shows the thinned result. Finally, Fig. 9.21(l) shows the thinned set converted to m-connectivity (see Section 2.5.2) to eliminate multiple paths.

9.5.6 Thickening

Thickening is the morphological dual of thinning and is defined by the expression

$$A \odot B = A \cup (A \circledast B) \tag{9.5-9}$$



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

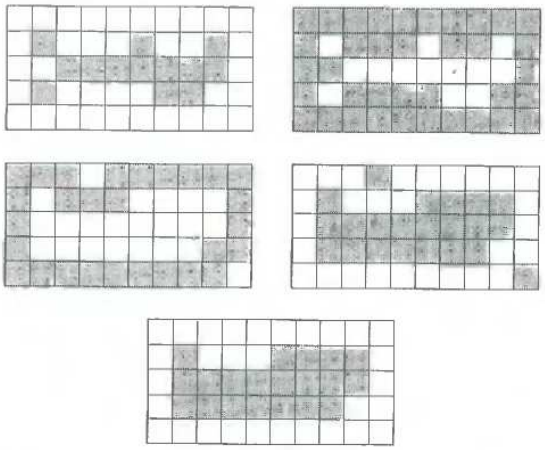
FIGURE 9.21 (a) Sequence of rotated structuring elements used for thinning. (b) Set A. (c) Result of thinning with the first element. (d)–(i) Results of thinning with the next seven elements (there was no change between the seventh and eighth elements). (j) Result of using the first element again (there were no changes for the next two elements). (k) Result after convergence. (l) Conversion to m-connectivity.

where B is a structuring element suitable for thickening. As in thinning, thickening can be defined as a sequential operation:

$$A \odot \{B\} = ((\ldots((A \odot B^1) \odot B^2)\ldots) \odot B^n). \tag{9.5-10}$$

The structuring elements used for thickening have the same form as those shown in Fig. 9.21(a) in connection with thinning, but with all 1's and 0's interchanged. However, a separate algorithm for thickening is seldom used in practice. Instead, the usual procedure is to thin the background of the set in question and then complement the result. In other words, to thicken a set A, we form $C = A^c$, thin C, and then form C^c. Figure 9.22 illustrates this procedure.

Depending on the nature of A, this procedure may result in some disconnected points, as Fig. 9.22(d) shows. Hence thickening by this method usually is followed by a simple postprocessing step to remove disconnected points. Note from Fig. 9.22(c) that the thinned background forms a boundary for the thick-



a b
c d
e

FIGURE 9.22 (a) Set A. (b) Complement of A. (c) Result of thinning the complement of A. (d) Thickened set obtained by complementing (c). (e) Final result, with no disconnected points.

ening process. This useful feature is not present in the direct implementation of thickening using Eq. (9.5-10), and it is one of the principal reasons for using background thinning to accomplish thickening.

9.5.7 Skeletons

As shown in Fig. 9.23, the notion of a skeleton, $S(A)$, of a set A is intuitively simple. We deduce from this figure that

(a) If z is a point of $S(A)$ and $(D)_z$ is the largest disk centered at z and contained in A, one cannot find a larger disk (not necessarily centered at z) containing $(D)_z$ and included in A. The disk $(D)_z$ is called a *maximum disk*.

(b) The disk $(D)_z$ touches the boundary of A at two or more different places.

The skeleton of A can be expressed in terms of erosions and openings. That is, it can be shown (Serra [1982]) that

$$S(A) = \bigcup_{k=0}^{K} S_k(A) \tag{9.5-11}$$

with

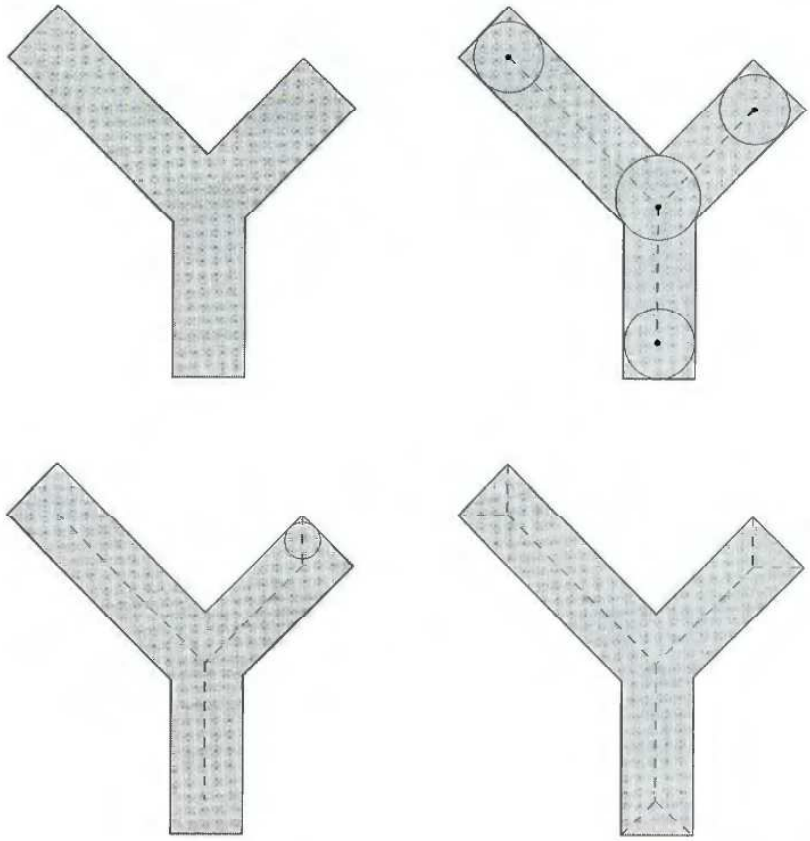
$$S_k(A) = (A \ominus kB) - (A \ominus kB) \circ B \tag{9.5-12}$$

where B is a structuring element, and $(A \ominus kB)$ indicates k successive erosions of A:

$$(A \ominus kB) = (\ldots(A \ominus B) \ominus B) \ominus \ldots) \ominus B \tag{9.5-13}$$

k times, and K is the last iterative step before A erodes to an empty set. In other words,

$$K = \max\{k \mid (A \ominus kB) \neq \varnothing\}. \tag{9.5-14}$$



a b
c d

FIGURE 9.23 (a) Set A. (b) Various positions of maximum disks with centers on the skeleton of A. (c) Another maximum disk on a different segment of the skeleton of A. (d) Complete skeleton.

The formulation given in Eqs. (9.5-11) and (9.5-12) states that $S(A)$ can be obtained as the union of the *skeleton subsets* $S_k(A)$. Also, it can be shown that A can be *reconstructed* from these subsets by using the equation

$$A = \bigcup_{k=0}^{K} (S_k(A) \oplus kB) \tag{9.5-15}$$

where $(S_k(A) \oplus kB)$ denotes k successive dilations of $S_k(A)$; that is,

$$(S_k(A) \oplus kB) = ((\ldots(S_k(A) \oplus B) \oplus B) \oplus \ldots) \oplus B. \tag{9.5-16}$$

EXAMPLE 9.8: Computing the skeleton of a simple figure.

■ Figure 9.24 illustrates the concepts just discussed. The first column shows the original set (at the top) and two erosions by the structuring element B. Note that one more erosion of A would yield the empty set, so $K = 2$ in this case. The second column shows the opening of the sets in the first column by B. These results are easily explained by the fitting characterization of the opening operation discussed in connection with Fig. 9.8. The third column simply contains the set differences between the first and second columns.

The fourth column contains two partial skeletons and the final result (at the bottom of the column). The final skeleton not only is thicker than it needs to be but, more important, it is not connected. This result is not unexpected, as nothing in the preceding formulation of the morphological skeleton guarantees connectivity. Morphology produces an elegant formulation in terms of erosions and openings of the given set. However, heuristic formulations such as the algo-

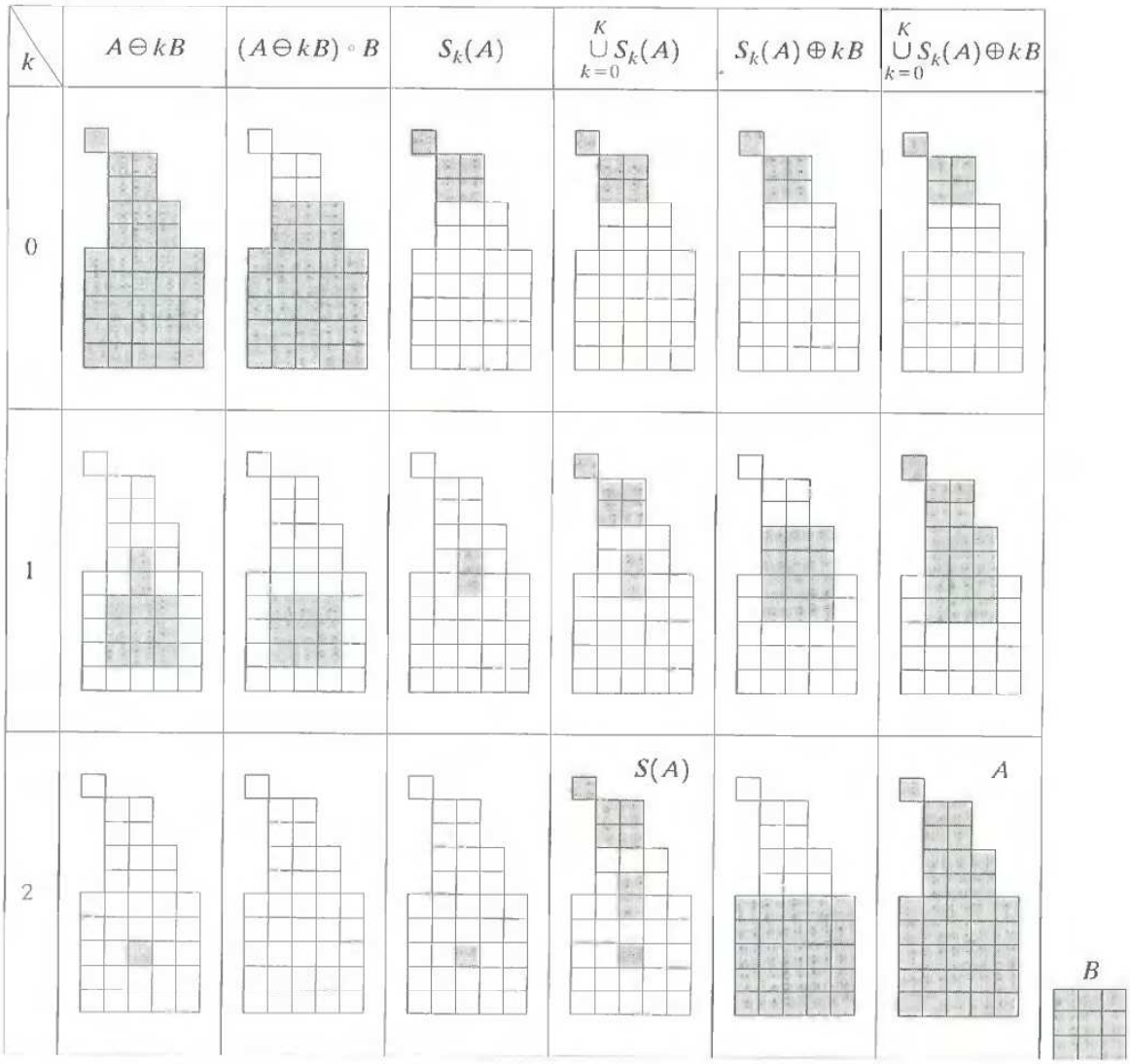


FIGURE 9.24 Implementation of Eqs. (9.5-11) through (9.5-15). The original set is at the top left, and its morphological skeleton is at the bottom of the fourth column. The reconstructed set is at the bottom of the sixth column.

rithm developed in Section 11.1.5 are needed if, as is usually the case, the skeleton must be maximally thin, connected, and minimally eroded.

The fifth column shows $S_0(A)$, $S_1(A) \oplus B$, and $(S_2(A) \oplus 2B) = (S_2(A) \oplus B) \oplus B$. Finally, the last column shows reconstruction of set A, which, according to Eq. (9.5-15), is the union of the dilated skeleton subsets shown in the fifth column. ■

9.5.8 Pruning

Pruning methods are an essential complement to thinning and skeletonizing algorithms because these procedures tend to leave parasitic components that need to be "cleaned up" by postprocessing. We begin the discussion with a pruning problem and then develop a morphological solution based on the material introduced in the preceding sections. Thus we take this opportunity to illustrate

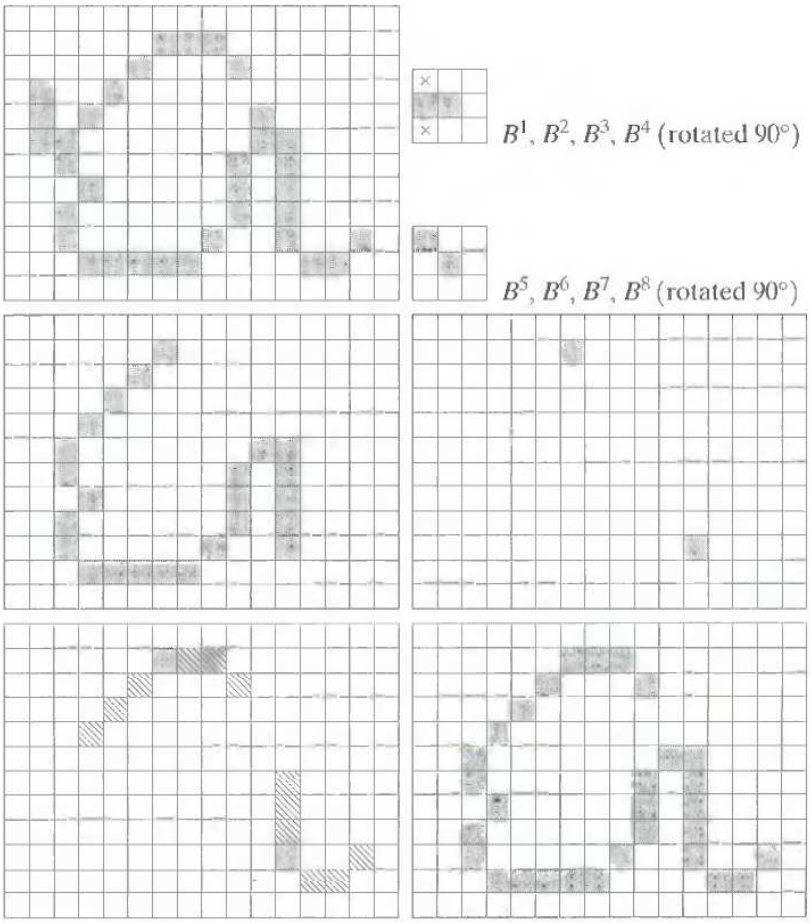
how to go about solving a problem by combining several of the techniques discussed up to this point.

A common approach in the automated recognition of hand-printed characters is to analyze the shape of the skeleton of each character. These skeletons often are characterized by "spurs" (parasitic components). Spurs are caused during erosion by non uniformities in the strokes composing the characters. We develop a morphological technique for handling this problem, starting with the assumption that the length of a parasitic component does not exceed a specified number of pixels.

Figure 9.25(a) shows the skeleton of a hand-printed "a." The parasitic component on the leftmost part of the character is illustrative of what we are interested in removing. The solution is based on suppressing a parasitic branch by successively eliminating its end point. Of course, this also shortens (or eliminates) other branches in the character but, in the absence of other structural information, the assumption in this example is that any branch with three or less pixels is to be eliminated. Thinning of an input set A with a sequence of structuring elements designed to detect only end points achieves the desired result. That is, let

$$X_1 = A \otimes \{B\} \tag{9.5-17}$$

where $\{B\}$ denotes the structuring element sequence shown in Figs. 9.25(b) and (c) [see Eq. (9.5-7) regarding structuring-element sequences]. The sequence of



a b
c
d e
f g

FIGURE 9.25 (a) Original image. (b) and (c) Structuring elements used for deleting end points. (d) Result of three cycles of thinning. (e) End points of (d). (f) Dilation of end points conditioned on (a). (g) Pruned image.

structuring elements consists of two different structures, each of which is rotated 90° for a total of eight elements. The × in Fig. 9.25(b) signifies a "don't care" condition, in the sense that it does not matter whether the pixel in that location has a value of 0 or 1. Numerous results reported in the literature on morphology are based on the use of a *single* structuring element, similar to the one in Fig. 9.25(b), but having "don't care" conditions along the entire first column. This is incorrect. For example, this element would identify the point located in the eighth row, fourth column of Fig. 9.25(a) as an end point, thus eliminating it and breaking connectivity in the stroke.

Applying Eq. (9.5-17) to A three times yields the set X_1 shown in Fig. 9.25(d). The next step is to "restore" the character to its original form, but with the parasitic branches removed. To do so first requires forming a set X_2 containing all end points in X_1 [Fig. 9.25(e)]:

$$X_2 = \bigcup_{k=1}^{8} (X_1 \circledast B^k) \tag{9.5-18}$$

where the B^k are the same end-point detectors shown in Figs. 9.25(b) and (c). The next step is dilation of the end points three times, using set A as a delimiter:

$$X_3 = (X_2 \oplus H) \cap A \tag{9.5-19}$$

where H is a 3×3 structuring element of 1's. As in the case of region filling and extraction of connected components, this type of conditional dilation prevents the creation of 1-valued elements outside the region of interest, as evidenced by the result shown in Fig. 9.25(f). Finally, the union of X_3 and X_1 yields the desired result,

$$X_4 = X_1 \cup X_3 \tag{9.5-20}$$

as shown in Fig. 9.25(g).

In more complex scenarios, use of Eq. (9.5-19) sometimes picks up the "tips" of some parasitic branches. This condition can occur when the end points of these branches are near the skeleton. Although Eq. (9.5-17) may eliminate them, they can be picked up again during dilation because they are valid points in A. Unless entire parasitic elements are picked up again (a rare case if these elements are short with respect to valid strokes), detecting and eliminating them is easy because they are disconnected regions.

A natural thought at this juncture is that there must be easier ways to solve this problem. For example, we could just keep track of all deleted points and simply reconnect the appropriate points to all end points left after application of Eq. (9.5-17). This option is valid, but the advantage of the formulation just presented is that the use of simple morphological constructs solved the entire problem. In practical situations when a set of such tools is available, the advantage is that no new algorithms have to be written. We simply combine the necessary morphological functions into a sequence of operations.

9.5.9 Summary of Morphological Operations on Binary Images

Table 9.2 summarizes the morphological results developed in the preceding sections, and Fig. 9.26 summarizes the basic types of structuring elements used in the various morphological processes discussed thus far. Note that the Roman numerals in the third column of Table 9.2 refer to the structuring elements in Fig. 9.26.

TABLE 9.2
Summary of morphological operations and their properties.

Operation	Equation	Comments (The Roman numerals refer to the structuring elements shown in Fig. 9.26).
Translation	$(A)_z = \{w \mid w = a + z, \quad \text{for } a \in A\}$	Translates the origin of A to point z.
Reflection	$\hat{B} = \{w \mid w = -b, \quad \text{for } b \in B\}$	Reflects all elements of B about the origin of this set.
Complement	$A^c = \{w \mid w \notin A\}$	Set of points not in A.
Difference	$A - B = \{w \mid w \in A, w \notin B\} = A \cap B^c$	Set of points that belong to A but not to B.
Dilation	$A \oplus B = \{z \mid (\hat{B})_z \cap A \neq \varnothing\}$	"Expands" the boundary of A. (I)
Erosion	$A \ominus B = \{z \mid (B)_z \subseteq A\}$	"Contracts" the boundary of A. (I)
Opening	$A \circ B = (A \ominus B) \oplus B$	Smoothes contours, breaks narrow isthmuses, and eliminates small islands and sharp peaks. (I)
Closing	$A \bullet B = (A \oplus B) \ominus B$	Smoothes contours, fuses narrow breaks and long thin gulfs, and eliminates small holes. (I)
Hit-or-miss transform	$A \circledast B = (A \ominus B_1) \cap (A^c \ominus B_2) = (A \ominus B_1) - (A \oplus \hat{B}_2)$	The set of points (coordinates) at which, simultaneously, B_1 found a match ("hit") in A and B_2 found a match in A^c.
Boundary extraction	$\beta(A) = A - (A \ominus B)$	Set of points on the boundary of set A. (I)
Region filling	$X_k = (X_{k-1} \oplus B) \cap A^c$; $X_0 = p$ and $k = 1, 2, 3, \ldots$	Fills a region in A, given a point p in the region. (II)
Connected components	$X_k = (X_{k-1} \oplus B) \cap A$; $X_0 = p$ and $k = 1, 2, 3, \ldots$	Finds a connected component Y in A, given a point p in Y. (I)
Convex hull	$X_k^i = (X_{k-1}^i \circledast B^i) \cup A$; $i = 1, 2, 3, 4$; $k = 1, 2, 3, \ldots$; $X_0^i = A$; and $D^i = X_{\text{conv}}^i$.	Finds the convex hull $C(A)$ of set A, where "conv" indicates convergence in the sense that $X_k^i = X_{k-1}^i$. (III)

TABLE 9.2 Summary of morphological results and their properties. *(continued)*

Operation	Equation	Comments (The Roman numerals refer to the structuring elements shown in Fig. 9.26).
Thinning	$A \otimes B = A - (A \circledast B)$ $= A \cap (A \circledast B)^c$ $A \otimes \{B\} =$ $((\ldots((A \otimes B^1) \otimes B^2)\ldots) \otimes B^n)$ $\{B\} = \{B^1, B^2, B^3, \ldots, B^n\}$	Thins set A. The first two equations give the basic definition of thinning. The last two equations denote thinning by a sequence of structuring elements. This method is normally used in practice. (IV)
Thickening	$A \odot B = A \cup (A \circledast B)$ $A \odot \{B\} =$ $((\ldots(A \odot B^1) \odot B^2 \ldots) \odot B^n)$	Thickens set A. (See preceding comments on sequences of structuring elements.) Uses IV with 0's and 1's reversed.
Skeletons	$S(A) = \bigcup_{k=0}^{K} S_k(A)$ $S_k(A) = \bigcup_{k=0}^{K} \{(A \ominus kB)$ $- [(A \ominus kB) \circ B]\}$ Reconstruction of A: $A = \bigcup_{k=0}^{K} (S_k(A) \oplus kB)$	Finds the skeleton $S(A)$ of set A. The last equation indicates that A can be reconstructed from its skeleton subsets $S_k(A)$. In all three equations, K is the value of the iterative step after which the set A erodes to the empty set. The notation $(A \ominus kB)$ denotes the kth iteration of successive erosion of A by B. (I)
Pruning	$X_1 = A \otimes \{B\}$ $X_2 = \bigcup_{k=1}^{8} (X_1 \circledast B^k)$ $X_3 = (X_2 \oplus H) \cap A$ $X_4 = X_1 \cup X_3$	X_4 is the result of pruning set A. The number of times that the first equation is applied to obtain X_1 must be specified. Structuring elements V are used for the first two equations. In the third equation H denotes structuring element I.

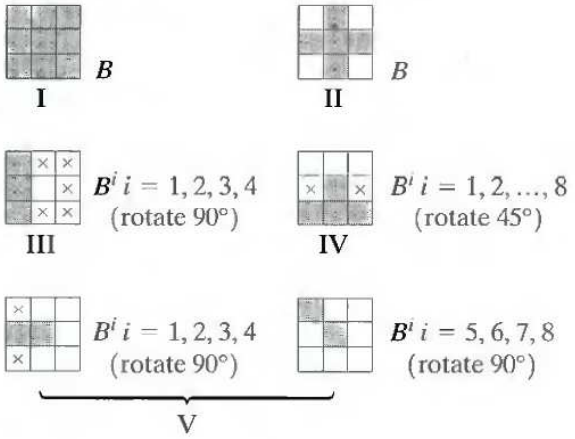


FIGURE 9.26 Five basic types of structuring elements used for binary morphology. The origin of each element is at its center and the ×'s indicate "don't care" values.

9.6 Extensions to Gray-Scale Images

In this section we extend to gray-scale images the basic operations of dilation, erosion, opening, and closing. We then use these operations to develop several basic gray-scale morphological algorithms. In particular, we develop algorithms for boundary extraction via a morphological gradient operation, and for region partitioning based on texture content. We also discuss algorithms for smoothing and sharpening, which often are useful as pre- or postprocessing steps.

Throughout the discussions that follow, we deal with digital image functions of the form $f(x, y)$ and $b(x, y)$, where $f(x, y)$ is the input image and $b(x, y)$ is a structuring element, itself a subimage function. The assumption is that these functions are discrete in the sense introduced in Section 2.4.2. That is, if Z denotes the set of real integers, the assumption is that (x, y) are integers from $Z \times Z$ and that f and b are functions that assign a gray-level value (a real number from the set of real numbers, R) to each distinct pair of coordinates (x, y). If the gray levels also are integers, Z replaces R.

9.6.1 Dilation

Gray-scale dilation of f by b, denoted $f \oplus b$, is defined as

$$(f \oplus b)(s, t) = \max\{f(s - x, t - y) + b(x, y) \mid (s - x), (t - y) \in D_f; (x, y) \in D_b\} \quad (9.6\text{-}1)$$

where D_f and D_b are the domains of f and b, respectively. Keep in mind that f and b are functions, rather than sets, as is the case in binary morphology.

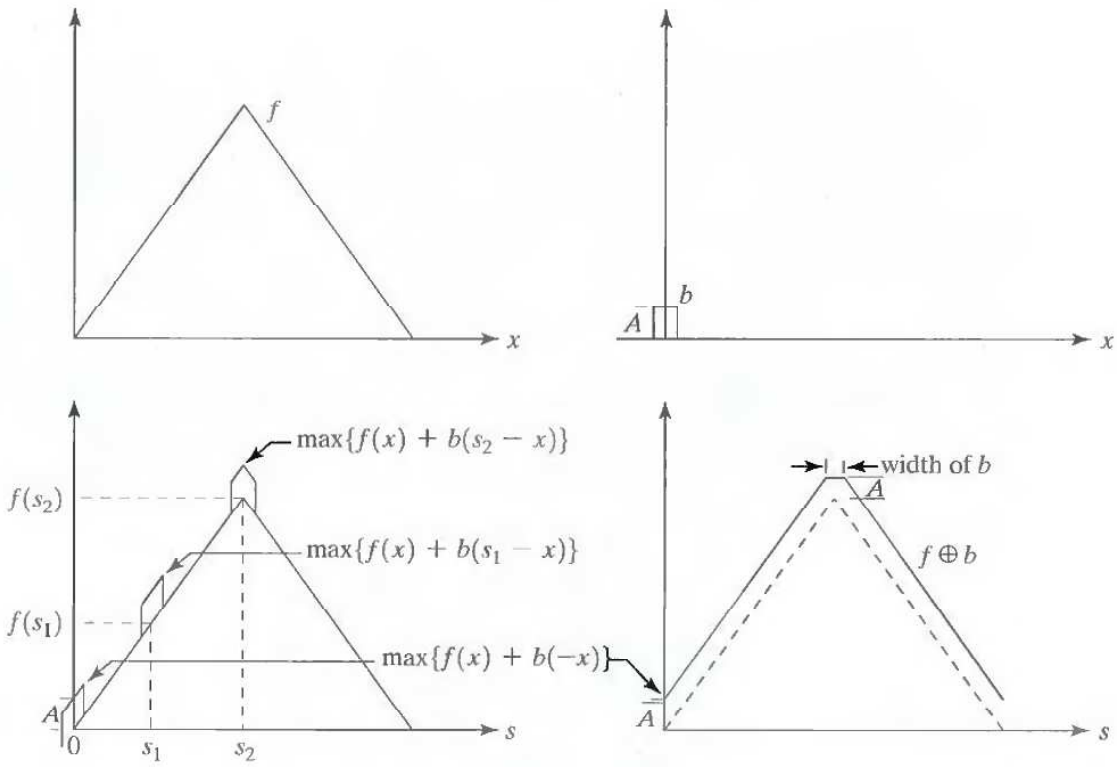
The condition that $(s - x)$ and $(t - y)$ have to be in the domain of f, and x and y have to be in the domain of b, is analogous to the condition in the binary definition of dilation, where the two sets have to overlap by at least one element. Note also that the form of Eq. (9.6-1) is similar to 2-D convolution [Eq. (4.2-30)], with the max operation replacing the sums of convolution and the addition replacing the products of convolution.

We illustrate the notation and mechanics of Eq. (9.6-1) by means of simple 1-D functions. For functions of one variable, Eq. (9.6-1) reduces to the expression

$$(f \oplus b)(s) = \max\{f(s-x) + b(x) \mid (s-x) \in D_f \quad \text{and} \quad x \in D_b\}.$$

Recall from the discussion of convolution that $f(-x)$ is simply $f(x)$ mirrored with respect to the origin of the x axis. As in convolution, the function $f(s-x)$ moves to the right for positive s, and to the left for negative s. The requirements that the value of $(s-x)$ has to be in the domain of f and that the value of x has to be in the domain of b imply that f and b overlap. As noted in the previous paragraph, these conditions are analogous to the requirement in the binary definition of dilation, where the two sets have to overlap by at least one element. Finally, unlike the binary case, f, rather than the structuring element b, is shifted. Equation (9.6-1) could be written so that b undergoes translation instead of f. However, if D_b is smaller than D_f (a condition almost always found in practice), the form given in Eq. (9.6-1) is simpler in terms of indexing and achieves the same result. Conceptually, f sliding by b is really no different than b sliding by f. In fact, although this equation is easier to implement, the actual mechanics of gray-scale dilation are easier to visualize if b is the function that slides past f.

An example is shown in Fig. 9.27. Note that at each position of the structuring element the value of dilation at that point is the maximum of the sum of f



a b
c d

FIGURE 9.27 (a) A simple function. (b) Structuring element of height A. (c) Result of dilation for various positions of sliding b past f. (d) Complete result of dilation (shown solid).

and b in the interval spanned by b. The general effect of performing dilation on a gray-scale image is twofold: (1) If all the values of the structuring element are positive, the output image tends to be brighter than the input. (2) Dark details either are reduced or eliminated, depending on how their values and shapes relate to the structuring element used for dilation.

9.6.2 Erosion

Gray-scale erosion, denoted $f \ominus b$, is defined as

$$(f \ominus b)(s,t) = \min\{f(s+x, t+y) - b(x,y) \mid (s+x), (t+y) \in D_f; (x,y) \in D_b\} \quad (9.6\text{-}2)$$

where D_f and D_b are the domains of f and b, respectively. The condition that $(s + x)$ and $(t + y)$ have to be in the domain of f, and x and y have to be in the domain of b, is analogous to the condition in the binary definition of erosion, where the structuring element has to be completely contained by the set being eroded. Note that the form of Eq. (9.6-2) is similar in form to 2-D correlation [Eq. (4.6-30)], with the min operation replacing the sums of correlation and subtraction replacing the products of correlation.

We illustrate the mechanics of Eq. (9.6-2) by eroding a simple 1-D function. For functions of one variable, the expression for erosion reduces to

$$(f \ominus b)(s) = \min\{f(s+x) - b(x) \mid (s+x) \in D_f \text{ and } x \in D_b\}.$$

As in correlation, the function $f(s + x)$ moves to the left for positive s and to the right for negative s. The requirements that $(s + x) \in D_f$ and $x \in D_b$ imply that the range of b is completely contained within the range of the displaced f. As noted in the previous paragraph, these requirements are analogous to those in the binary definition of erosion, where the structuring element has to be contained completely in the set being eroded.

Finally, unlike the binary definition of erosion, f, rather than the structuring element b, is shifted. Equation (9.6-2) could be written so that b is the function translated, resulting in a more complicated expression in terms of indexing. Because f sliding past b conceptually is the same as b sliding past f, the form of Eq. (9.6-2) is used for the reasons stated at the end of the discussion on dilation. Figure 9.28 shows the result of eroding the function of Fig. 9.27(a) by the structuring element of Fig. 9.27(b).

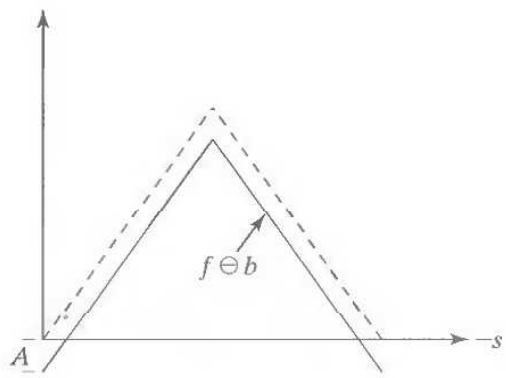


FIGURE 9.28 Erosion of the function shown in Fig. 9.27(a) by the structuring element shown in Fig. 9.27(b).

Equation (9.6-2) indicates that erosion is based on choosing the minimum value of $(f - b)$ in the interval defined by the shape of the structuring element. The general effect of performing erosion on a gray-scale image is twofold: (1) If all the elements of the structuring element are positive, the output image tends to be darker than the input image. (2) The effect of bright details in the input image that are smaller in area than the structuring element is reduced, with the degree of reduction being determined by the gray-level values surrounding the bright detail and by the shape and amplitude values of the structuring element itself.

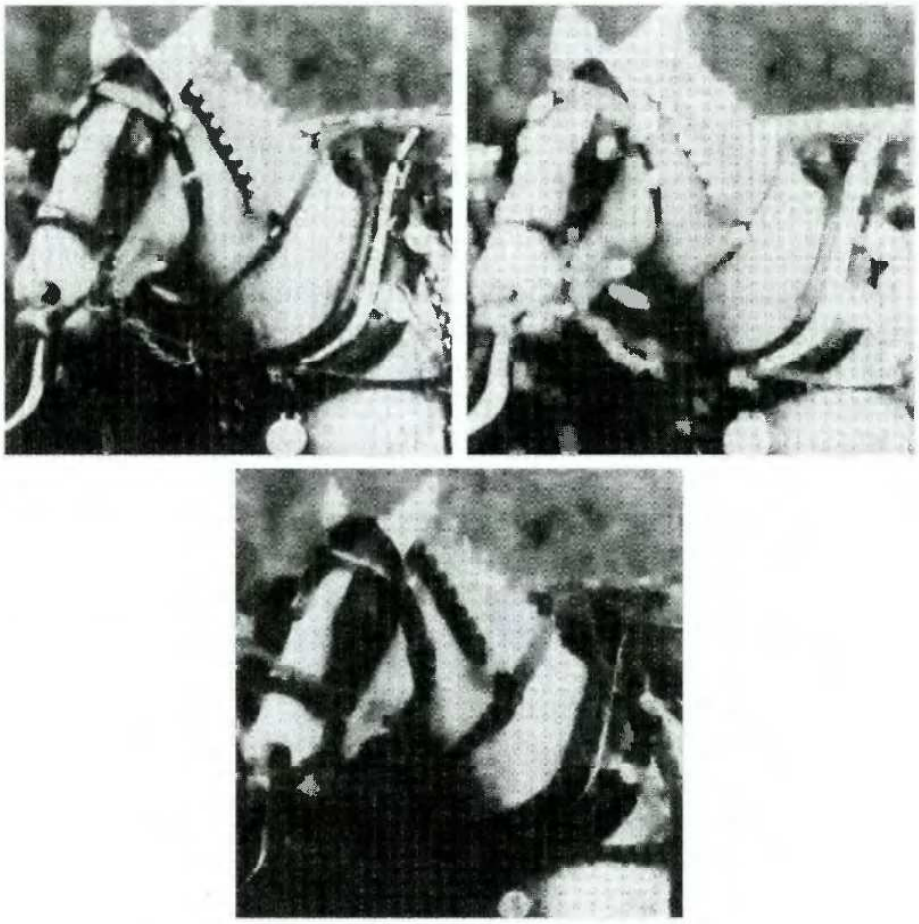
Gray-scale dilation and erosion are duals with respect to function complementation and reflection. That is,

$$(f \ominus b)^c(s,t) = (f^c \oplus \hat{b})(s,t) \tag{9.6-3}$$

where $f^c = -f(x, y)$ and $\hat{b} = b(-x, -y)$. Except as needed for clarity, we simplify the notation in the following discussions by omitting the arguments of all functions.

EXAMPLE 9.9: Illustration of dilation and erosion on a gray-scale image.

■ Figure 9.29(a) shows a simple 512×512 gray-scale image, and Fig. 9.29(b) shows the result of dilating this image with a "flat-top" structuring element in the shape of a parallelepiped of unit height and size 5×5 pixels. Based on the preceding discussion, dilation is expected to produce an image that is brighter than the original and in which small, dark details have been reduced or eliminated.



a b
c

FIGURE 9.29 (a) Original image. (b) Result of dilation. (c) Result of erosion. (Courtesy of Mr. A. Morris, Leica Cambridge, Ltd.)

These effects clearly are visible in Fig. 9.29(b). Not only does the image appear brighter than the original, but the sizes of dark features, such as the nostrils and the dark components of the studded rein extending from the ears down to the neck, have been reduced. Figure 9.29(c) shows the result of eroding the original image. Note the opposite effect to dilation. The eroded image is darker, and the sizes of small, bright features (such as the studs on the rein) were reduced. ■

9.6.3 Opening and Closing

The expressions for opening and closing of gray-scale images have the same form as their binary counterparts. The opening of image f by subimage (structuring element) b, denoted $f \circ b$, is

$$f \circ b = (f \ominus b) \oplus b. \tag{9.6-4}$$

As in the binary case, opening is simply the erosion of f by b, followed by a dilation of the result by b. Similarly, the closing of f by b, denoted $f \bullet b$, is

$$f \bullet b = (f \oplus b) \ominus b. \tag{9.6-5}$$

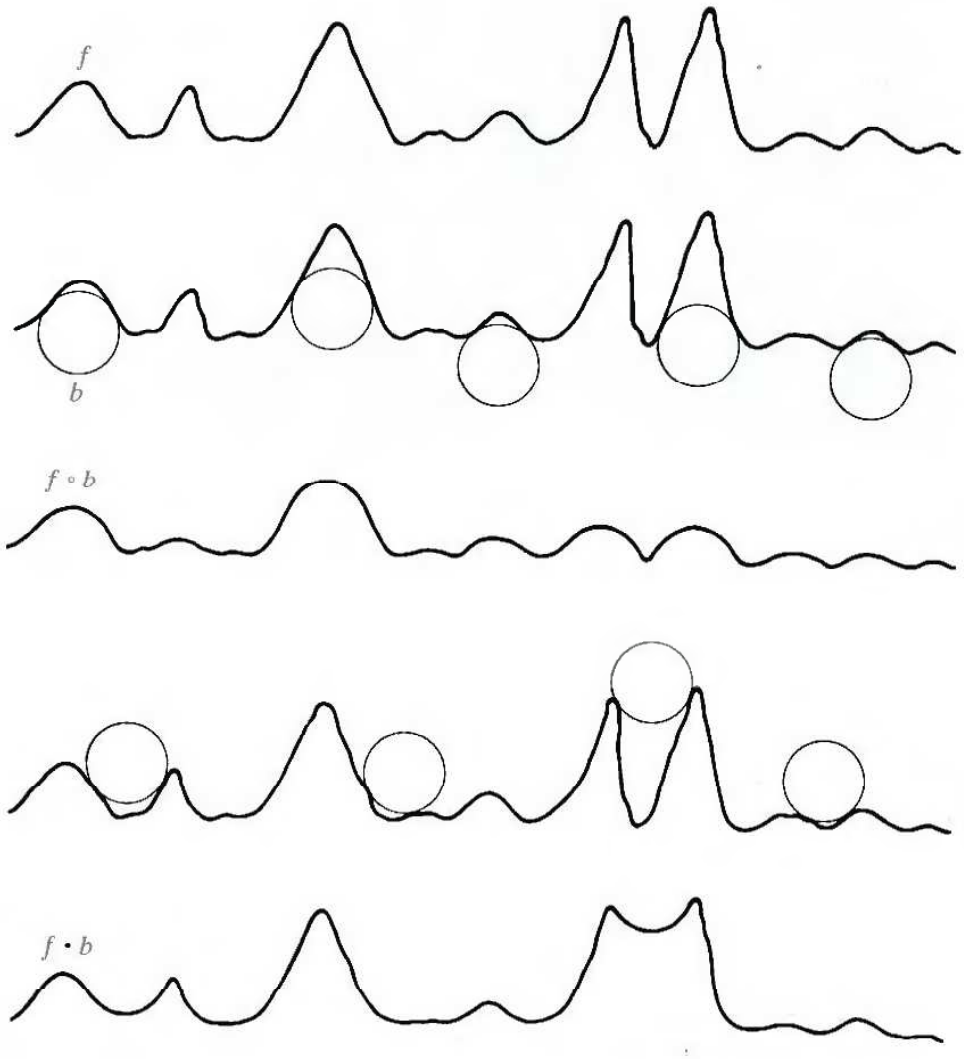
The opening and closing for gray-scale images are duals with respect to complementation and reflection. That is,

$$(f \bullet b)^c = f^c \circ \hat{b}. \tag{9.6-6}$$

Because $f^c = -f(x, y)$, Eq. (9.6-6) can be written also as $-(f \bullet b) = (-f \circ \hat{b})$.

Opening and closing of images have a simple geometric interpretation. Suppose that we view an image function $f(x, y)$ in 3-D perspective (like a relief map), with the x- and y-axes being the usual spatial coordinates and the third axis being gray-level values. In this representation, the image appears as a discrete surface whose value at any point (x, y) is the value of f at those coordinates. Suppose that we open f by a spherical structuring element, b, viewing this element as a "rolling ball." Then the mechanics of opening f by b may be interpreted geometrically as the process of pushing the ball against the *underside* of the surface, while at the same time rolling it so that the entire underside of the surface is traversed. The opening, $f \circ b$, then is the surface of the highest points reached by any part of the sphere as it slides over the *entire* undersurface of f.

Figure 9.30 illustrates this concept. Figure 9.30(a) shows a scan line of a gray-scale image as a continuous function to simplify the illustration. Figure 9.30(b) shows the rolling ball in various positions, and Fig. 9.30(c) shows the complete result of opening f by b along the scan line. All the peaks that were narrow with respect to the diameter of the ball were reduced in amplitude and sharpness. In practical applications, opening operations usually are applied to remove small (with respect to the size of the structuring element) light details, while leaving the overall gray levels and larger bright features relatively undisturbed. The initial erosion removes the small details, but it also darkens the image. The subsequent dilation again increases the overall intensity of the image without reintroducing the details totally removed by erosion.



a
b
c
d
e

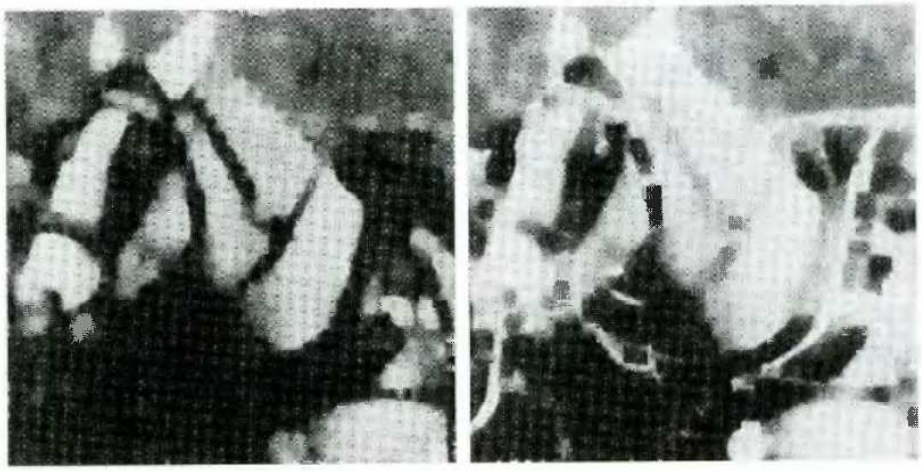
FIGURE 9.30 (a) A gray-scale scan line. (b) Positions of rolling ball for opening. (c) Result of opening. (d) Positions of rolling ball for closing. (e) Result of closing.

Figures 9.30(d) and (e) show the result of closing f by b. Here, the ball slides on top of the surface, and peaks essentially are left in their original form (assuming that their separation at the narrowest point exceeds the diameter of the ball). In practice, closing is generally used to remove dark details from an image, while leaving bright features relatively undisturbed. The initial dilation removes the dark details and brightens the image, and the subsequent erosion darkens the image without reintroducing the details removed by dilation. It is of interest to compare Fig. 9.30 with Figs. 9.8 and 9.9.

The gray-scale opening operation satisfies the following properties:

(i) $(f \circ b) \lrcorner f$.
(ii) If $f_1 \lrcorner f_2$, then $(f_1 \circ b) \lrcorner (f_2 \circ b)$.
(iii) $(f \circ b) \circ b = f \circ b$.

The notation $e \lrcorner r$ is used to indicate that the domain of e is a subset of the domain of r, and also that $e(x, y) \leq r(x, y)$ for any (x, y) in the domain of e.



a b

FIGURE 9.31 (a) Opening and (b) closing of Fig. 9.29(a). (Courtesy of Mr. A. Morris, Leica Cambridge, Ltd.)

Similarly, the closing operation satisfies the following properties:

(i) $f \lrcorner (f \bullet b)$.
(ii) If $f_1 \lrcorner f_2$, then $(f_1 \bullet b) \lrcorner (f_2 \bullet b)$.
(iii) $(f \bullet b) \bullet b = f \bullet b$.

The usefulness of these expressions is similar to that of their binary counterparts.

EXAMPLE 9.10: Illustration of gray-scale opening and closing.

■ Figure 9.31(a) shows the result of opening the image in Fig. 9.29(a) with the same structuring element used there. Note the decreased sizes of the small, bright details, with no appreciable effect on the darker gray levels. Figure 9.31(b) shows the closing of Fig. 9.29(a). Note the decreased sizes of the small, dark details, with relatively little effect on the bright features. ■

9.6.4 Some Applications of Gray-Scale Morphology

We conclude the discussion of morphological techniques by presenting in some detail various applications of gray-scale morphology. Unless stated otherwise, all the images shown are of size 512 × 512 and were processed by using the structuring element discussed in connection with Fig. 9.29.

Morphological smoothing

One way to achieve smoothing is to perform a morphological opening followed by a closing. The net result of these two operations is to remove or attenuate both bright and dark artifacts or noise. Figure 9.32 shows a smoothed version of the image shown in Fig. 9.29(a).

Morphological gradient

In addition to the operations discussed earlier in connection with the removal of small dark and bright artifacts, dilation and erosion often are used to com-

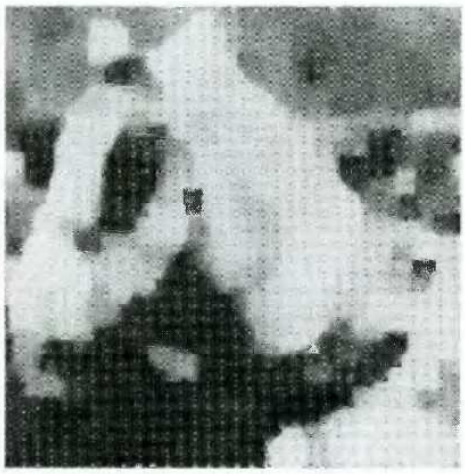


FIGURE 9.32 Morphological smoothing of the image in Fig. 9.29(a). (Courtesy of Mr. A. Morris, Leica Cambridge, Ltd.)

pute the *morphological gradient* of an image, denoted g:

$$g = (f \oplus b) - (f \ominus b). \tag{9.6-7}$$

Figure 9.33 shows the result of computing the morphological gradient of the image shown in Fig. 9.29(a). As expected, the morphological gradient highlights sharp gray-level transitions in the input image. As opposed to gradients obtained using the methods discussed in Section 3.7.3, morphological gradients obtained using symmetrical structuring elements tend to depend less on edge directionality.

Top-hat transformation

The so-called morphological *top-hat* transformation of an image, denoted h, is defined as

$$h = f - (f \circ b) \tag{9.6-8}$$

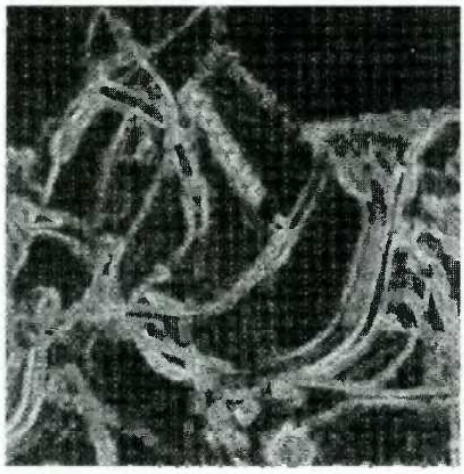


FIGURE 9.33 Morphological gradient of the image in Fig. 9.29(a). (Courtesy of Mr. A. Morris, Leica Cambridge, Ltd.)

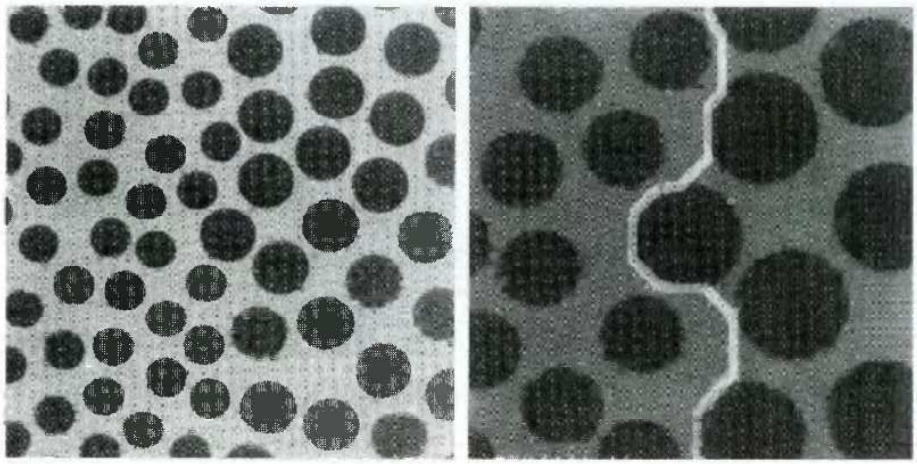
FIGURE 9.34 Result of performing a top-hat transformation on the image of Fig. 9.29(a). (Courtesy of Mr. A. Morris, Leica Cambridge, Ltd.)

where, as before, f is the input image and b is the structuring element function. This transformation—which owes its original name to the use of a cylindrical or parallelepiped structuring element function with a flat top—is useful for enhancing detail in the presence of shading. Figure 9.34 shows the result of performing a top-hat transformation on the image of Fig. 9.29(a). Note the enhancement of detail in the background region below the lower part of the horse's head.

Textural segmentation

Figure 9.35(a) shows a simple gray-scale image composed of two texture regions. The region on the right consists of circular blobs of larger diameter than those on the left. The objective is to find the boundary between the two regions based on their textural content.

Because closing tends to remove dark details from an image, the procedure in this particular case is to close the input image by using successively larger structuring elements. When the size of the structuring element corresponds to that of



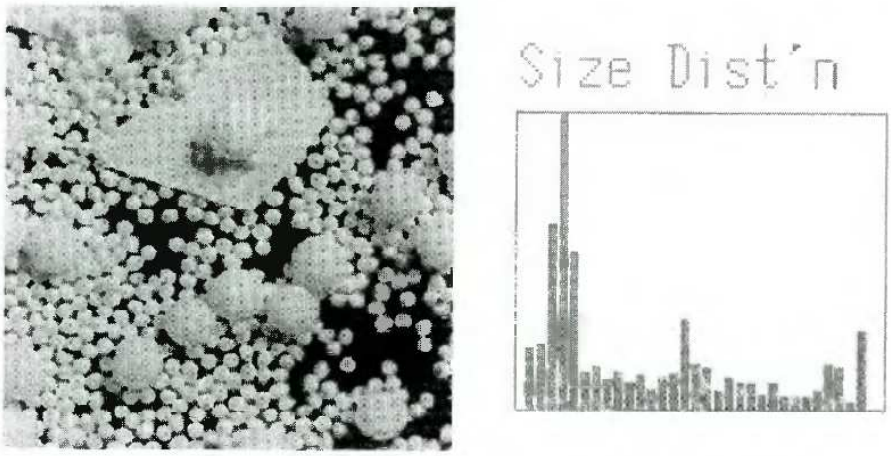
a b

FIGURE 9.35 (a) Original image. (b) Image showing boundary between regions of different texture. (Courtesy of Mr. A. Morris, Leica Cambridge, Ltd.)

the small blobs, they are removed from the image, leaving only a light background in the area previously occupied by them. At this point in the process, only the larger blobs and the light background on the left and between the large blobs themselves, remain. Next, a single opening is performed with a structuring element that is large in relation to the separation between the large blobs. This operation removes the light patches between the blobs, leaving a dark region on the right consisting of the large dark blobs and the now equally dark patches between these blobs. At this point the process has produced a light region on the left and a dark region on the right. A simple threshold then yields the boundary between the two textural regions. Figure 9.35(b) shows the resulting boundary superimposed on the original image. It is instructive to work through this example in more detail using the rolling ball analogy described in Fig. 9.30.

Granulometry

Granulometry is a field that deals principally with determining the size distribution of particles in an image. Figure 9.36(a) shows an image consisting of light objects of three different sizes. The objects not only are overlapping, but they also are too cluttered to enable detection of individual particles. Because the particles are lighter than the background, the following morphological approach can be used to determine size distribution. Opening operations with structuring elements of increasing size are performed on the original image. The difference between the original image and its opening is computed after each pass when a different structuring element is completed. At the end of the process, these differences are normalized and then used to construct a histogram of particle-size distribution. This approach is based on the idea that opening operations of a particular size have the most effect on regions of the input image that contain particles of similar size. Thus, a measure of the relative number of such particles is obtained by computing the difference between the input and output images. Figure 9.36(b) shows the resulting size distribution in this case. The histogram indicates the presence of three predominant particle sizes in the input image. This type of processing is useful for describing regions with a predominant particle-like character.



a b

FIGURE 9.36 (a) Original image consisting of overlapping particles; (b) size distribution. (Courtesy of Mr. A. Morris, Leica Cambridge, Ltd.)

Summary

The morphological concepts and techniques introduced in this chapter constitute a powerful set of tools for extracting features of interest in an image. One of the most appealing aspects of morphological image processing is the extensive set-theoretical foundation from which morphological techniques have evolved. A significant advantage in terms of implementation is the fact that dilation and erosion are primitive operations that are the basis for a broad class of morphological algorithms. As shown in the following chapter, morphology can be used as the basis for developing image segmentation procedures with a wide range of applications. As discussed in Chapter 11, morphological techniques also play a major role in procedures for image description.

References and Further Reading

The book by Serra [1982] is a fundamental reference on morphological image processing. See also Serra [1988], Giardina and Dougherty [1988], and Haralick and Shapiro [1992]. Additional early references relevant to our discussion include Blum [1967], Lantuéjoul [1980], Maragos [1987], and Haralick et al. [1987]. For an overview of both binary and gray-scale morphology, see Basart and Gonzalez [1992] and Basart et al. [1992]. This set of references provides ample background for the material covered in Sections 9.1 through 9.4.

Important issues of implementing morphological algorithms such as the ones given in Section 9.5 and 9.6 are exemplified in the papers by Jones and Svalbe [1994], Park and Chin [1995], Sussner and Ritter [1997], Anelli et al. [1998], and Shaked and Bruckstein [1998]. Current work in the theory and applications of morphological image processing is summarized in the book by Goutsias and Bloomberg [2000], and in a special issue of *Pattern Recognition* [2000]. See also a recent compilation of references by Rosenfeld [2000]. The books by Marchand-Maillet and Sharaiha [2000] on binary image processing, and by Ritter and Wilson [2001] on image algebra, also are on interest.

Problems

See inside front cover

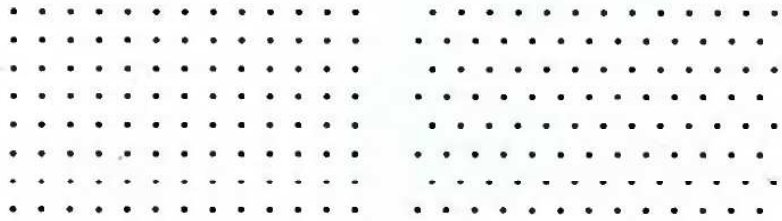
Detailed solutions to the problems marked with a star can be found in the book web site. The site also contains suggested projects based on the material in this chapter.

9.1 Digital images in this book are embedded in square grid arrangements and pixels are allowed to be 4-, 8-, or m-connected. However, other grid arrangements are possible. Specifically, a hexagonal grid arrangement that leads to 6-connectivity, is sometimes used (see the following figure).

(a) How would you convert an image from a square grid to a hexagonal grid?

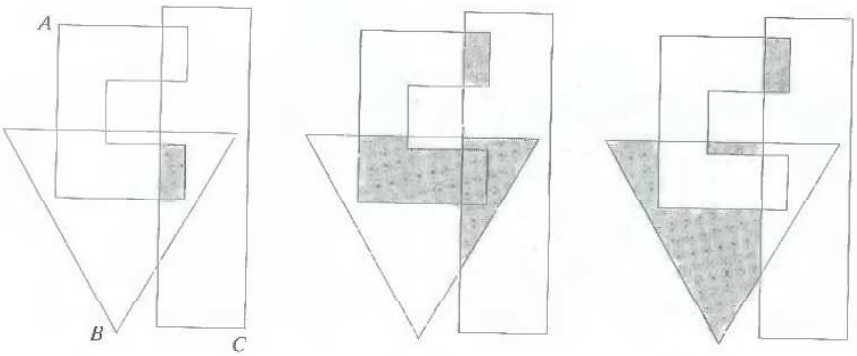
(b) Discuss the shape invariance to rotation of objects represented in a square grid as opposed to a hexagonal grid.

(c) Is it possible to have ambiguous diagonal configurations in a hexagonal grid, as is the case with 8-connectivity? (See Section 2.5.2.)



9.2 ★ **(a)** With reference to Fig. 9.1(a), sketch the set $(A \cap B) \cup (A \cup B)^c$.

(b) Give expressions for the sets shown shaded in the following figures. The shaded areas in each image constitute one set, so give an expression for each of the three images:



9.3 ★ **(a)** Give a morphological algorithm for converting an 8-connected binary boundary to an m-connected boundary (see Section 2.5.2). You may assume that the boundary is fully connected and that it is one pixel thick.

(b) Does the operation of your algorithm require more than one iteration with each structuring element? Explain your reasoning.

(c) Is the performance of your algorithm independent of the order in which the structuring elements are applied? If your answer is yes, prove it; otherwise give an example that illustrates the dependence of your procedure on the order of application of the structuring elements.

9.4 Erosion of a set A by structuring element B is a subset of A as long as the origin of B is contained by B. Give an example in which the erosion $A \ominus B$ lies outside, or partially outside, A.

9.5 The following four statements are true. Advance an argument that establishes the reason(s) for their validity. Part (a) is true in general. Parts (b) through (d) are true only for *digital* sets. To show the validity of (b) through (d), draw a discrete, square grid (as shown in Problem 9.1) and give an example for each case using sets composed of points on this grid. *Hint:* Keep the number of points in each case as small as possible while still establishing the validity of the statements.

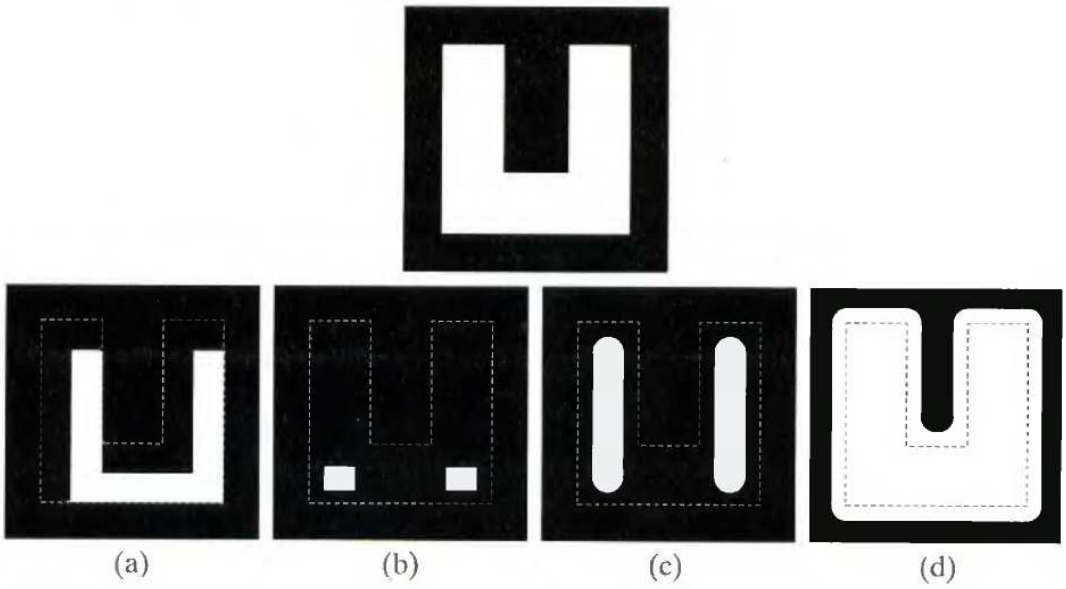
★ **(a)** The erosion of a convex set by a convex structuring element is a convex set.

★ **(b)** The dilation of a convex set by a convex structuring element is not necessarily always convex.

(c) The points in a convex digital set are not always connected.

(d) It is possible to have a set of points in which a line joining every pair of points in the set lies within the set but the set is not convex.

★**9.6** With reference to the image shown, give the structuring element and morphological operation(s) that produced each of the results shown in images (a) through (d). Show the origin of each structuring element clearly. The dashed lines show the boundary of the original set and are included only for reference. Note that in (d) all corners are rounded.



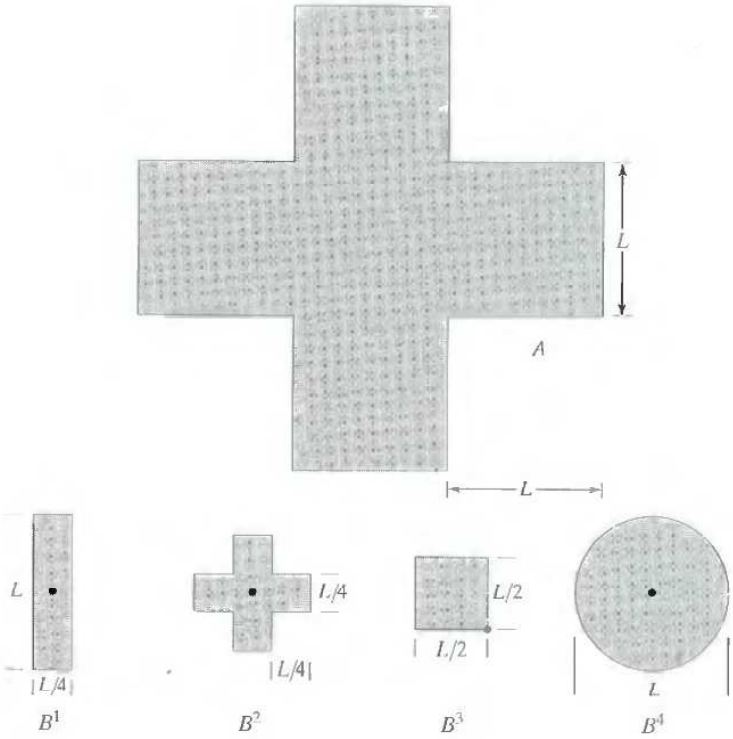
9.7 Let A denote the set shown shaded in the following figure. Refer to the structuring elements shown (the black dots denote the origin). Sketch the result of the following morphological operations:

(a) $(A \ominus B^4) \oplus B^2$

(b) $(A \ominus B^1) \oplus B^3$

(c) $(A \oplus B^1) \oplus B^3$

(d) $(A \oplus B^3) \ominus B^2$



★**9.8** **(a)** What is the limiting effect of repeatedly dilating an image? Assume that a trivial (one point) structuring element is not used.

(b) What is the smallest image from which you can start in order for your answer in part (a) to hold?

9.9 **(a)** What is the limiting effect of repeatedly eroding an image? Assume that a trivial (one point) structuring element is not used.

(b) What is the smallest image from which you can start in order for your answer in part (a) to hold?

★**9.10** An alternative definition of dilation is

$$A \oplus B = \{w \in Z^2 \mid w = a + b, \text{ for some } a \in A \text{ and } b \in B\}.$$

Show that this definition and the definition in Eq. (9.2-1) are equivalent.

9.11 **(a)** Show that the definition of dilation given in Problem 9.10 is equivalent to yet another definition of dilation:

$$A \oplus B = \bigcup_{b \in B} (A)_b$$

(This expression also is called the *Minkowsky addition* of two sets.)

(b) Show that the expression in (a) also is equivalent to the definition in Eq. (9.2-1).

★**9.12** An alternative definition of erosion is

$$A \ominus B = \{w \in Z^2 \mid w + b \in A, \text{ for every } b \in B\}.$$

Show that this definition is equivalent to the definition in Eq. (9.2-3).

9.13 **(a)** Show that the definition of erosion given in Problem 9.12 is equivalent to yet another definition of erosion:

$$A \ominus B = \bigcap_{b \in B} (A)_{-b}$$

(If $-b$ is replaced with b, this expression is called the *Minkowsky subtraction* of two sets.)

(b) Show that the expression in (a) also is equivalent to the definition in Eq. (9.2-3).

★**9.14** Prove the validity of the duality expression $(A \bullet B)^c = (A^c \circ \hat{B})$.

9.15 Prove the validity of the following expressions:

★ **(a)** $A \circ B$ is a subset (subimage) of A.

(b) If C is a subset of D, then $C \circ B$ is a subset of $D \circ B$.

(c) $(A \circ B) \circ B = A \circ B$.

9.16 Prove the validity of the following expressions (assume that the origin of B is contained in B and that Problem 9.16(a) is true):

(a) A is a subset (subimage) of $A \bullet B$.

(b) If C is a subset of D, then $C \bullet B$ is a subset of $D \bullet B$.

(c) $(A \bullet B) \bullet B = A \bullet B$.

9.17 Refer to the image and structuring element shown. Sketch what the sets C, D, E, and F would look like in the following sequence of operations: $C = A \ominus B$; $D = C \oplus B$; $E = D \oplus B$; and $F = E \ominus B$. The initial set A consists of all the

image components shown in white, with the exception of the structuring element B. Note that this sequence of operations is simply the opening of A by B, followed by the closing of that opening by B. You may assume that B is just large enough to enclose each of the noise components.

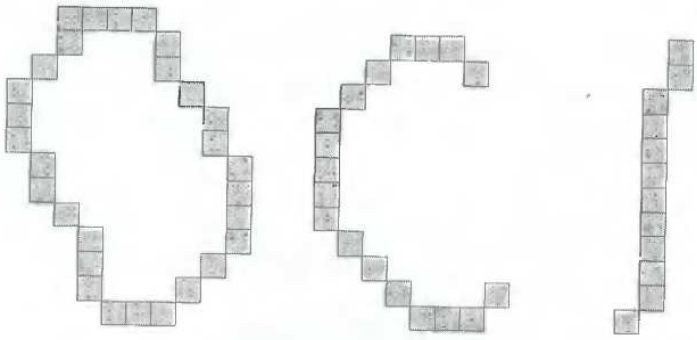
★**9.18** In the example given in connection with Fig. 9.7, it is demonstrated that the squares surviving the erosion step could be fully restored by dilating them with the same structuring element used for erosion. Erosion followed by dilation is the opening of the image, and we know that, in general, opening does not result in identical reconstruction of the objects to which it is applied. Explain why full reconstruction of the remaining squares was possible in Fig. 9.7.

9.19 **(a)** Sketch the result of applying the hit-or-miss transform to the image and structuring element shown. Indicate clearly the origin and border you selected for the structuring element.

(b) Compare your result with the result of matching by correlation (Fig. 4.46), stating differences and similarities between the two.

Image Structuring element

★**9.20** Three features (lake, bay, and line segment) useful for differentiating thinned objects in an image are shown in the following figures. Develop a morphological/logical algorithm for differentiating among these shapes. The input to your algorithm would be one of these three shapes. The output must be the identity of the input. You may assume that the features are 1 pixel thick and that each is fully connected. However, they can appear in any orientation.



Lake Bay Line segment

9.21 Discuss what you would expect the result to be in each of the following cases:

(a) The starting point of the region filling algorithm of section 9.5.2 is a point on the boundary of the object.

(b) The starting point in the region filling algorithm is outside of the boundary.

(c) Sketch what the convex hull of the figure in Problem 9.7 would look like as computed with the algorithm given in Section 9.5.4. Assume that $L = 3$ pixels.

★9.22 **(a)** Discuss the effect of using the structuring element in Fig. 9.15(c) for boundary extraction, instead of the one shown in Fig. 9.13(b).

(b) What would be the effect of using a 3×3 structuring element composed of all 1's in the region filling algorithm of Eq. (9.5-2), instead of the structuring element shown in Fig. 9.15(c)?

9.23 Propose a method for fully automating the example shown in Fig. 9.16. This implies determining which black points are background points and which are points contained within the spheres (i.e., black regions completely contained within white regions). Assume that binary 1's are represented in white.

★9.24 The algorithm given in Section 9.5.3 for extracting connected components requires that a point be known in each connected component in order to extract them all. Suppose that you are given a binary image containing an arbitrary (unknown) number of connected components. Propose a completely automated procedure for extracting all connected components. Assume that points belonging to connected components are labeled 1 and background points are labeled 0.

9.25 Suppose that the image $f(x, y)$ and structuring element $b(x, y)$ in Eq. (9.6-1) both are rectangular, with domains D_f and D_b denoted $([F_{x1}, F_{x2}], [F_{y1}, F_{y2}])$ and $([B_{x1}, B_{x2}], [B_{y1}, B_{y2}])$ respectively. For example, the closed intervals $[F_{x1}, F_{x2}]$ and $[F_{y1}, F_{y2}]$ are the ranges of x and y in the x- and y-axes of the xy-plane where the function $f(x, y)$ is defined.

★ (a) Assume that $(x, y) \in D_b$, and derive expressions for the intervals over which the displacement variables s and t can range to satisfy Eq. (9.6-1). These intervals in the s- and t-axes define the rectangular domain of $(f \oplus b)(s, t)$ in the st-plane.

(b) Repeat for erosion, as defined in Eq. (9.6-2).

9.26 A gray-scale image, $f(x, y)$, is corrupted by nonoverlapping noise spikes that can be modeled as small, cylindrical artifacts of radii $R_{\min} \leq r \leq R_{\max}$ and amplitude $A_{\min} \leq a \leq A_{\max}$.

★ (a) Develop a morphological filtering approach for cleaning up the image.

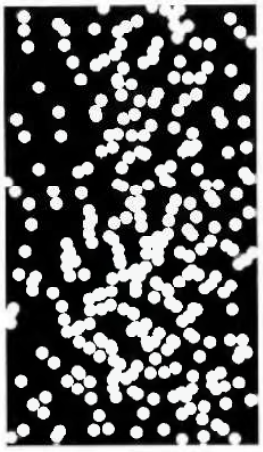
(b) Repeat (a), but now assume that there is overlapping of, at most, four noise spikes.

9.27 A preprocessing step in an application of microscopy is concerned with the issue of isolating individual round particles from similar particles that overlap in groups of two or more particles (see following image). Assuming that all particles are of the same size, propose a morphological algorithm that produces three images consisting respectively of

★ **(a)** Only of particles that have merged with the boundary of the image.

(b) Only overlapping particles.

(c) Only nonoverlapping particles.



9.28 A high-technology manufacturing plant wins a government contract to manufacture high-precision washers of the form shown in the following figure. The terms of the contract require that the shape of all washers be inspected by an imaging system. In this context, shape inspection refers to deviations from round on the inner and outer edges of the washers. You may assume the following: (1) A "golden" (perfect with respect to the problem) image of an acceptable washer is available; and (2) the imaging and positioning systems ultimately used in the system will have an accuracy high enough to allow you to ignore errors due to digitalization and positioning. You are hired as a consultant to help specify the visual inspection part of the system. Propose a solution based on morphological/logic operations. Your answer should be in the form of a block diagram.

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 input and output 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 to which the subdivision is carried depends on the problem being solved. That is, segmentation should stop when the objects of interest in an application have been isolated. For example, in the automated inspection of electronic assemblies, interest lies in analyzing images of the 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 rugged segmentation. In some situations, such as industrial inspection applications, at least some measure of control over the environment is possible at times. 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 of the environment. Then 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.

Image segmentation algorithms generally 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 in an image. 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 a number of approaches in the two categories just mentioned. We begin the development with methods suitable for detecting gray-level discontinuities such as points, lines, and edges. Edge detection in particular has been a staple of segmentation algorithms for many years. In addition to edge detection per se, we also discuss methods for connecting edge segments and for "assembling" edges into region boundaries. The discussion on edge detection is followed by the introduction of various thresholding techniques. Thresholding also is a fundamental approach to segmentation that enjoys a significant degree of popularity, especially in applications where speed is an important factor. The discussion on thresholding is followed by the development of several region-oriented segmentation approaches. We then discuss a morphological approach to segmentation called *watershed segmentation*. 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 discussion on the use of motion cues for image segmentation.

10.1 Detection of Discontinuities

In this section we present several techniques for detecting the three basic types of gray-level discontinuities in a digital image: points, lines, and edges. The most common way to look for discontinuities is to run a mask through the image in the manner described in Section 3.5. For the 3×3 mask shown in Fig. 10.1, this procedure involves computing the sum of products of the coefficients with the gray

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

FIGURE 10.1 A general 3×3 mask.

levels contained in the region encompassed by the mask. That is, with reference to Eq. (3.5-3), the response of the mask at any point in the image is given by

$$R = w_1 z_1 + w_2 z_2 + \cdots + w_9 z_9 = \sum_{i=1}^{9} w_i z_i \tag{10.1-1}$$

where z_i is the gray level of the pixel associated with mask coefficient w_i. As usual, the response of the mask is defined with respect to its center location. The details for implementing mask operations are discussed in Section 3.5.

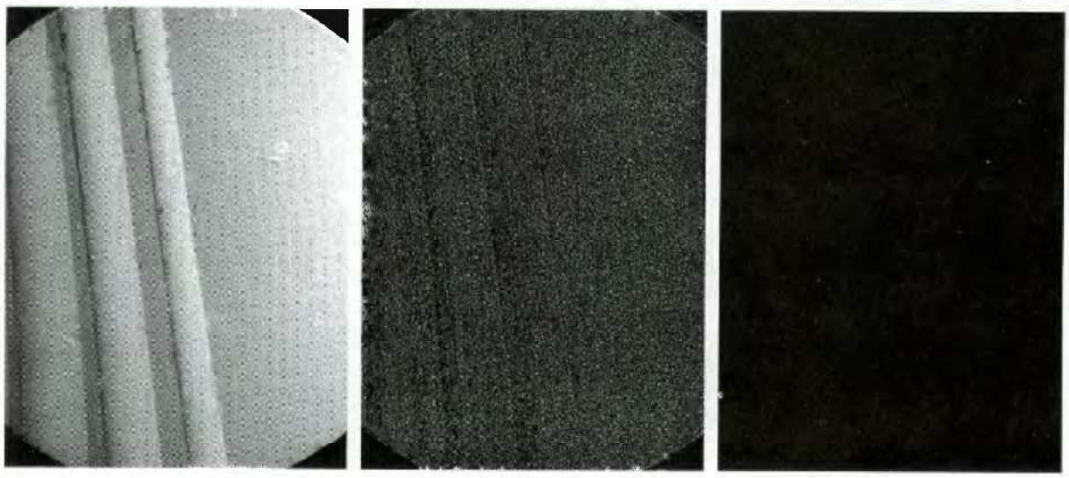
10.1.1 Point Detection

The detection of isolated points in an image is straightforward in principle. Using the mask shown in Fig. 10.2(a), we say that a point has been detected at the location on which the mask is centered if

$$|R| \geq T \tag{10.1-2}$$

where T is a nonnegative threshold and R is given by Eq. (10.1-1). Basically, this formulation measures the weighted differences between the center point and its neighbors. The idea is that an *isolated* point (a point whose gray level is significantly different from its background and which is located in a homogeneous or nearly homogeneous area) will be quite different from its surroundings, and thus be easily detectable by this type of mask. Note that the mask in Fig. 10.2(a) is the same as the mask shown in Fig. 3.39(d) in connection with Laplacian operations. However, the emphasis here is strictly on the detection of points. That is, the only differences that are considered of interest are those

−1	−1	−1
−1	8	−1
−1	−1	−1



a
b c d

FIGURE 10.2 (a) Point detection mask. (b) X-ray image of a turbine blade with a porosity. (c) Result of point detection. (d) Result of using Eq. (10.1-2). (Original image courtesy of X-TEK Systems Ltd.)

large enough (as determined by T) to be considered isolated points. Note that the mask coefficients sum to zero, indicating that the mask response will be zero in areas of constant gray level.

EXAMPLE 10.1: Detection of isolated points in an image.

■ We illustrate segmentation of isolated points from an image with the aid of Fig. 10.2(b), which shows an X-ray image of a jet-engine turbine blade with a porosity in the upper, right quadrant of the image. There is a single black pixel embedded within the porosity. Figure 10.2(c) is the result of applying the point detector mask to the X-ray image, and Fig. 10.2(d) shows the result of using Eq. (10.1-2) with T equal to 90% of the highest absolute pixel value of the image in Fig. 10.2(c). (Threshold selection is discussed in detail in Section 10.3.) The single pixel is clearly visible in this image (the pixel was enlarged manually so that it would be visible after printing). This type of detection process is rather specialized because it is based on single-pixel discontinuities that have 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 gray-level discontinuities. ■

10.1.2 Line Detection

The next level of complexity is line detection. Consider the masks shown in Fig. 10.3. If the first mask were moved around an image, it would respond more strongly to lines (one pixel thick) oriented horizontally. With a constant background, the maximum response would result when the line passed through the middle row of the mask. This is easily verified by sketching a simple array of 1's with a line of a different gray level (say, 5's) running horizontally through the array. A similar experiment would reveal that the second mask in Fig. 10.3 responds best to lines oriented at +45°; the third mask to vertical lines; and the fourth mask to lines in the −45° direction. These directions can be established also by noting that the preferred direction of each mask is weighted with a larger coefficient (i.e., 2) than other possible directions. Note that the coefficients in each mask sum to zero, indicating a zero response from the masks in areas of constant gray level.

Let R_1, R_2, R_3, and R_4 denote the responses of the masks in Fig. 10.3, from left to right, where the R's are given by Eq. (10.1-1). Suppose that the four masks are run individually through an image. If, at a certain point in the image, $|R_i| > |R_j|$, for all $j \neq i$, that point is said to be more likely associated with a line in the direction of mask i. For example, if at a point in the image, $|R_1| > |R_j|$ for

FIGURE 10.3 Line masks.

−1	−1	−1
2	2	2
−1	−1	−1

Horizontal

−1	−1	2
−1	2	−1
2	−1	−1

+45°

−1	2	−1
−1	2	−1
−1	2	−1

Vertical

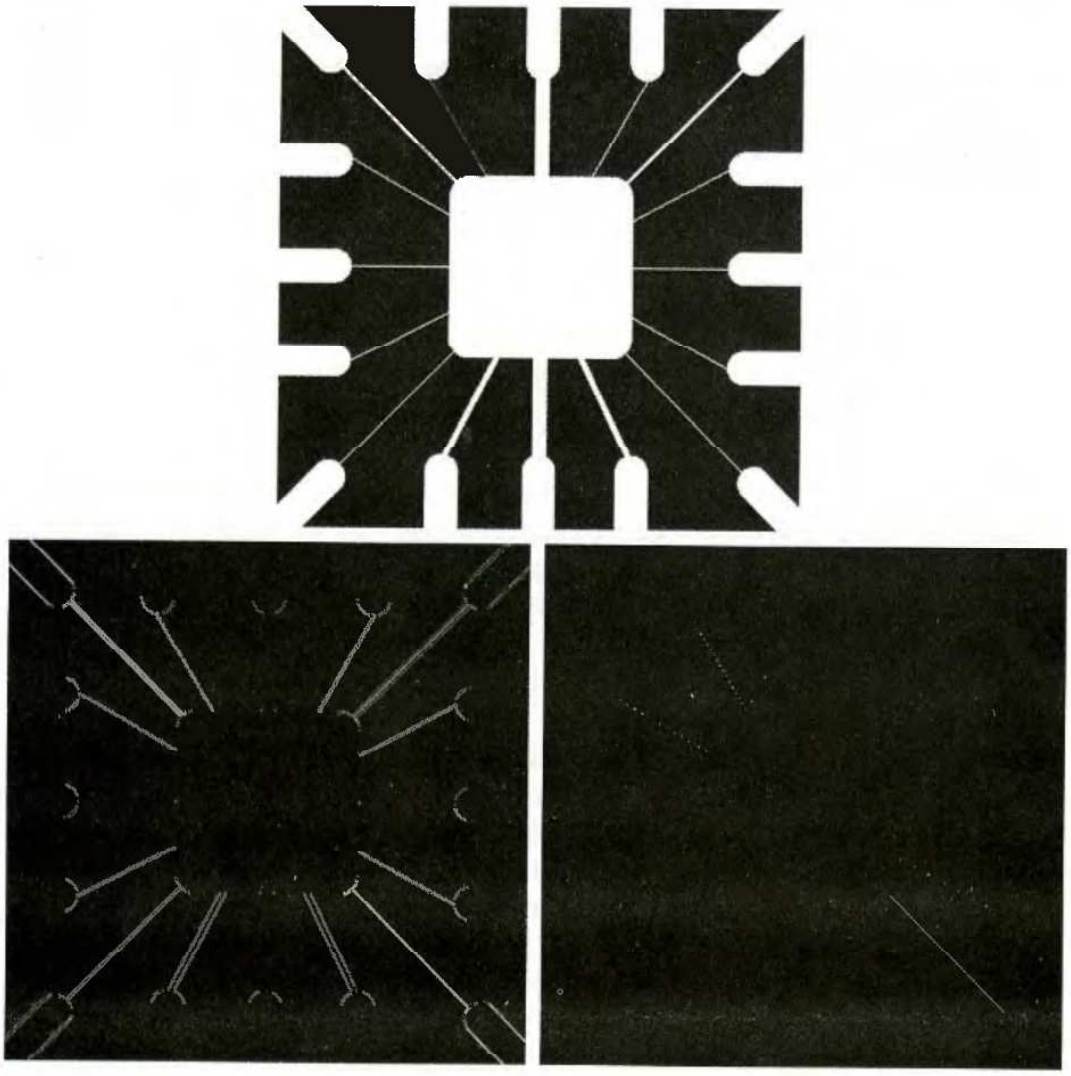
2	−1	−1
−1	2	−1
−1	−1	2

−45°

$j = 2, 3, 4$, 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. (10.1-2). 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 one pixel thick, correspond closest to the direction defined by the mask. The following example illustrates this procedure.

EXAMPLE 10.2: Detection of lines in a specified direction.

Figure 10.4(a) shows a digitized (binary) portion of a wire-bond mask for an electronic circuit. Suppose that we are interested in finding all the lines that are one pixel thick and are oriented at $-45°$. For this purpose, we use the last mask shown in Fig. 10.3. The absolute value of the result is shown in Fig. 10.4(b). Note that all vertical and horizontal components of the image were eliminated, and that the components of the original image that tend toward a $-45°$ direction



a
b c

FIGURE 10.4 Illustration of line detection. (a) Binary wire-bond mask. (b) Absolute value of result after processing with $-45°$ line detector. (c) Result of thresholding image (b).

produced the strongest responses in Fig. 10.4(b). In order to determine which lines best fit the mask, we simply threshold this image. The result of using a threshold equal to the maximum value in the image is shown in Fig. 10.4(c). The maximum value is a good choice for a threshold in applications such as this because the input image is binary and we are looking for the strongest responses. Figure 10.4(c) shows in white all points that passed the threshold test. In this case, the procedure extracted the only line segment that was one pixel thick and oriented at $-45°$ (the other component of the image oriented in this direction in the top, left quadrant is not one pixel thick). The isolated points shown in Fig. 10.4(c) 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 as way that the mask produced a maximum response at those isolated locations. These isolated points can be detected using the mask in Fig. 10.2(a) and then deleted, or they could be deleted using morphological erosion, as discussed in the last chapter. ■

10.1.3 Edge Detection

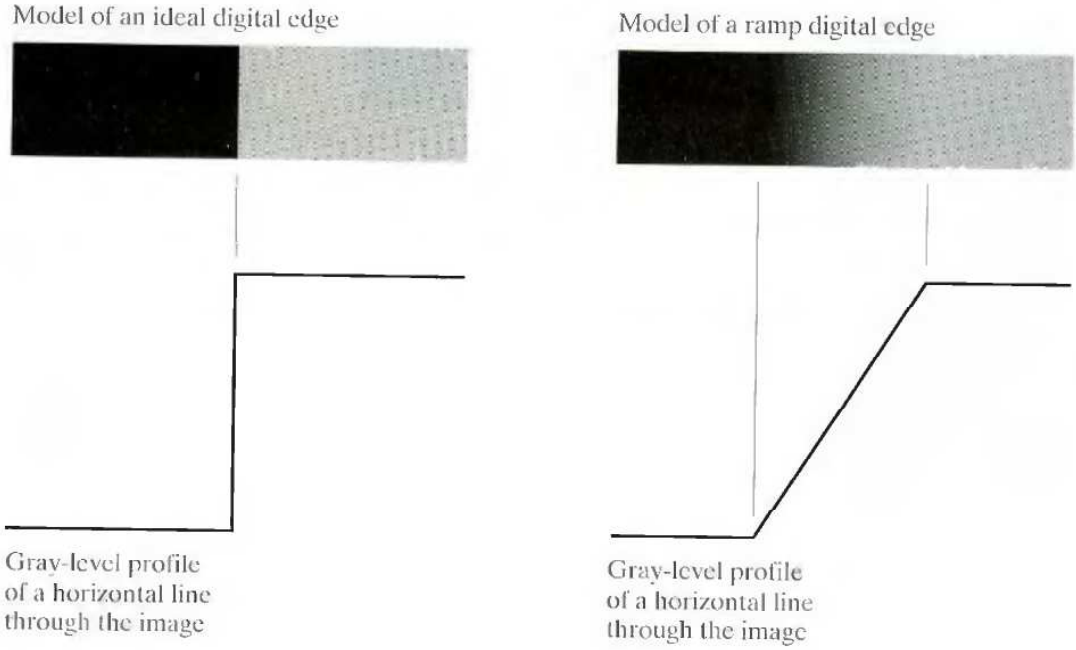
Although point and line detection certainly are important in any discussion on segmentation, edge detection is by far the most common approach for detecting meaningful discontinuities in gray level. In this section we discuss approaches for implementing first- and second-order digital derivatives for the detection of edges in an image. We introduced these derivatives in Section 3.7 in the context of image enhancement. The focus in this section is on their properties for edge detection. Some of the concepts previously introduced are restated briefly here for the sake continuity in the discussion.

Basic formulation

Edges were introduced informally in Section 3.7.1. In this section we look at the concept of a digital edge a little closer. Intuitively, an edge is a set of connected pixels that lie on the boundary between two regions. However, we already went through some length in Section 2.5.2 to explain the difference between an edge and a boundary. Fundamentally, as we shall see shortly, an edge is a "local" concept whereas a region boundary, owing to the way it is defined, is a more global idea. A reasonable definition of "edge" requires the ability to measure gray-level transitions in a meaningful way.

We start by modeling an edge intuitively. This will lead us to a formalism in which "meaningful" transitions in gray levels can be measured. Intuitively, an ideal edge has the properties of the model shown in Fig. 10.5(a). An ideal edge according to this model is a set of connected pixels (in the vertical direction here), each of which is located at an orthogonal step transition in gray level (as shown by the horizontal profile in the figure).

In practice, optics, sampling, and other image acquisition imperfections yield edges that are blurred, with the degree of blurring being determined by factors such as the quality of the image acquisition system, the sampling rate, and illumination conditions under which the image is acquired. As a result, edges are more closely modeled as having a "ramplike" profile, such as the one shown in



a b

FIGURE 10.5 (a) Model of an ideal digital edge. (b) Model of a ramp edge. The slope of the ramp is proportional to the degree of blurring in the edge.

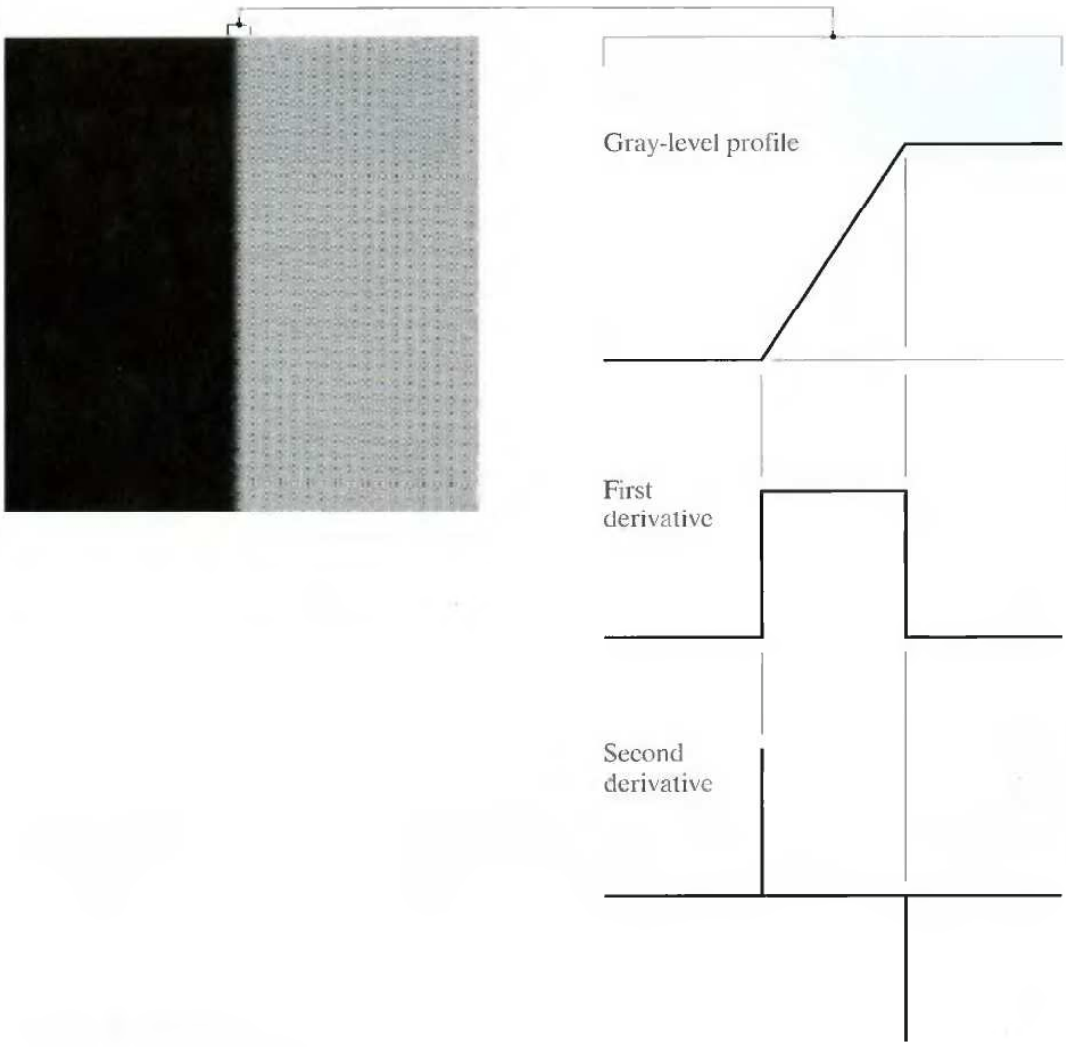
Fig. 10.5(b). The slope of the ramp is inversely proportional to the degree of blurring in the edge. In this model, we no longer have a thin (one pixel thick) path. Instead, an edge point now is any point contained in the ramp, and an edge would then be a set of such points that are connected. The "thickness" of the edge is determined by the length of the ramp, as it transitions from an initial to a final gray level. This length is determined by the slope, which, in turn, is determined by the degree of blurring. This makes sense: Blurred edges tend to be thick and sharp edges tend to be thin.

Figure 10.6(a) shows the image from which the close-up in Fig. 10.5(b) was extracted. Figure 10.6(b) shows a horizontal gray-level profile of the edge between the two regions. This figure also shows the first and second derivatives of the gray-level profile. The first derivative is positive at the points of transition into and out of the ramp as we move from left to right along the profile; it is constant for points in the ramp; and is zero in areas of constant gray level. The second derivative is positive at the transition associated with the dark side of the edge, negative at the transition associated with the light side of the edge, and zero along the ramp and in areas of constant gray level. The signs of the derivatives in Fig. 10.6(b) would be reversed for an edge that transitions from light to dark.

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 (i.e., to determine if a point is on a ramp). 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) an imaginary straight line joining the extreme positive and negative values of the second derivative would cross zero near the midpoint of the edge. This *zero-crossing* property of the second derivative is quite useful

a b

FIGURE 10.6 (a) Two regions separated by a vertical edge. (b) Detail near the edge, showing a gray-level profile, and the first and second derivatives of the profile.



for locating the centers of thick edges, as we show later in this section. Finally, we note that some edge models make use of a smooth transition into and out of the ramp (Problem 10.5). However, the conclusions at which we arrive in the following discussion are the same. Also, it is evident from this discussion that we are dealing here with local measures (thus the comment made in Section 2.5.2 about the local nature of edges).

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 as in the preceding discussion.

EXAMPLE 10.3: Behavior of the first and second derivatives around a noisy edge.

■ The edges shown in Fig. 10.5 and 10.6 are free of noise. The image segments in the first column in Fig. 10.7 show close-ups of four ramp edges separating a black region on the left and a white region on the right. It is important to 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 of Fig. 10.7 are corrupted by additive Gaussian noise with zero mean and

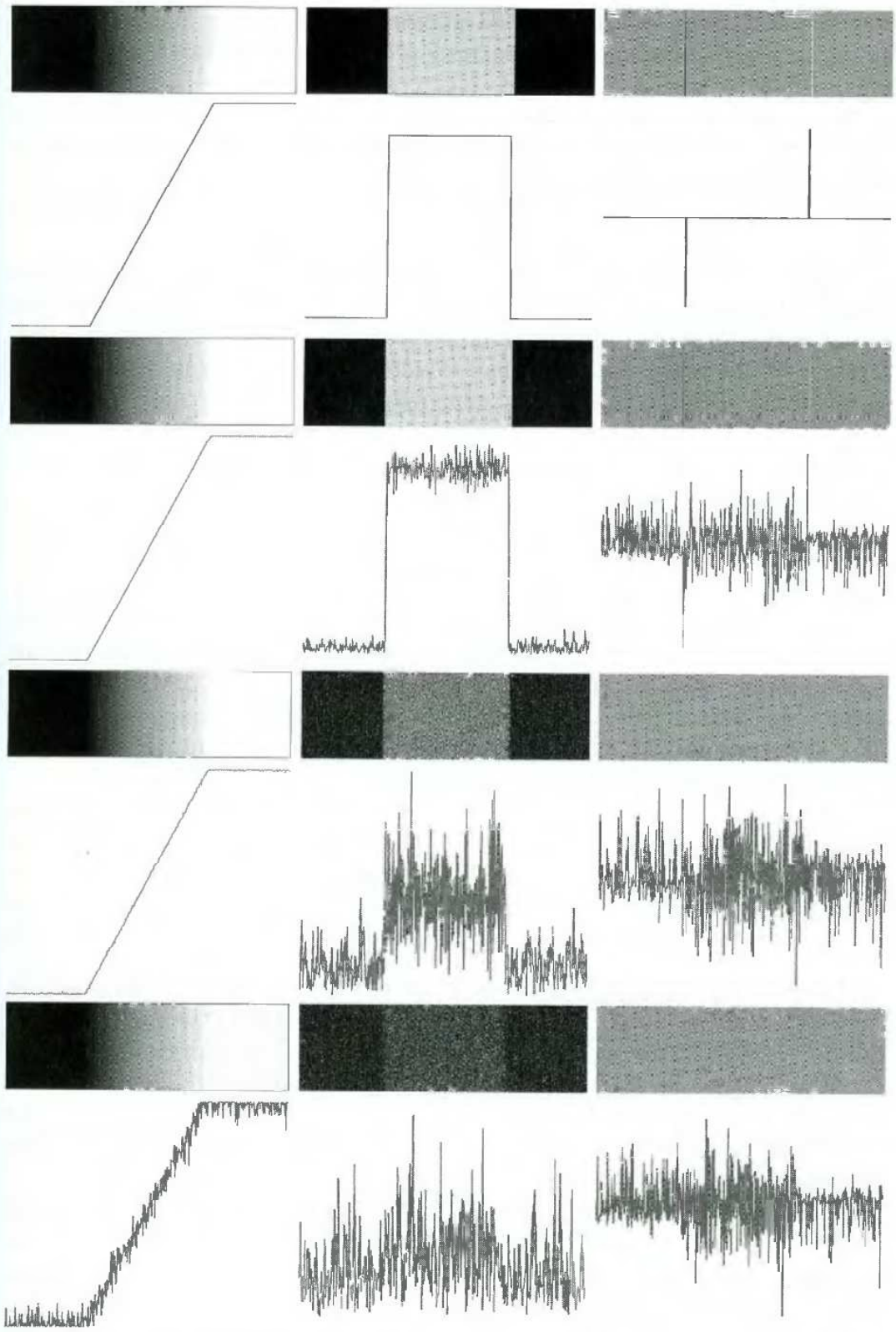


FIGURE 10.7 First column: images and gray-level profiles of a ramp edge corrupted by random Gaussian noise of mean 0 and $\sigma = 0.0, 0.1, 1.0$, and 10.0, respectively. Second column: first-derivative images and gray-level profiles. Third column: second-derivative images and gray-level profiles.

a
b
c
d

standard deviation of 0.1, 1.0, and 10.0 gray levels, respectively. The graph shown below each of these images is a gray-level profile of a horizontal scan line passing through the image.

The images in the second column of Fig. 10.7 are the first-order derivatives of the images on the left (we discuss computation of the first and second image derivatives in the following section). Consider, for example, the center image at the top. As discussed in connection with Fig. 10.6(b), the derivative is zero in the constant black and white regions. These are the two black areas shown in the derivative image. The derivative of a constant ramp is a constant, equal to the slope of the ramp. This constant area in the derivative image is 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 that column with a ramp edge. What makes these results interesting is that the noise really is almost invisible in the images on the left column. The last image is a slightly grainy, but this corruption is almost imperceptible. 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 in the top, right image. The thin black and white lines are the positive and negative components explained in Fig. 10.6. The gray in these images represents zero due to scaling. We note that the only noisy second derivative that resembles the noiseless case is the one corresponding to noise with a standard deviation of 0.1 gray levels. The other two 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 fairly little noise can have such a significant impact on the two key derivatives used for edge detection in images 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. ■

Based on this example and on the three paragraphs that precede it, we are led to the conclusion that, to be classified as a meaningful edge point, the transition in gray level associated with that point has to be significantly stronger than the background at that point. Since we are dealing with local computations, the method of choice to determine whether a value is "significant" or not is to use a threshold. Thus, we *define* a point in an image as being an *edge point* if its two-dimensional first-order derivative is greater than a specified threshold. A set of such points that are connected according to a predefined criterion of connectedness (see Section 2.5.2) is by definition an *edge*. The term *edge segment* generally is used if the edge is short in relation to the dimensions of the image. A key problem in segmentation is to assemble edge segments into longer edges, as explained in Section 10.2. An alternate definition if we elect to use the second-derivative is simply to define the edge points in an image as the zero crossings of its second derivative. The definition of an edge in this case is the same as above. It is important to note that these definitions do not guarantee success in finding edges in an image. They simply give us a formalism to look for them.

As in Chapter 3, first-order derivatives in an image are computed using the gradient. Second-order derivatives are obtained using the Laplacian.

Gradient operators

First-order derivatives of a digital image are based on various approximations of the 2-D gradient. The gradient of an image $f(x, y)$ at location (x, y) is defined as the *vector*

See inside front cover

Consult the book web site for a brief review of vector analysis.

$$\nabla \mathbf{f} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f}{\partial x} \\ \dfrac{\partial f}{\partial y} \end{bmatrix}. \tag{10.1-3}$$

It is well known from vector analysis that the gradient vector points in the direction of maximum rate of change of f at coordinates (x, y).

An important quantity in edge detection is the magnitude of this vector, denoted ∇f, where

$$\nabla f = \text{mag}(\nabla \mathbf{f}) = \left[G_x^2 + G_y^2\right]^{1/2}. \tag{10.1-4}$$

This quantity gives the maximum rate of increase of $f(x, y)$ per unit distance in the direction of $\nabla \mathbf{f}$. It is a common (although not strictly correct) practice to refer to ∇f also as the *gradient*. We will adhere to convention and also use this term interchangeably, differentiating between the vector and its magnitude only in cases in which confusion is likely.

The *direction* of the gradient vector also is an important quantity. Let $\alpha(x, y)$ represent the direction angle of the vector $\nabla \mathbf{f}$ at (x, y). Then, from vector analysis,

$$\alpha(x, y) = \tan^{-1}\left(\frac{G_y}{G_x}\right) \tag{10.1-5}$$

where the angle is measured with respect to the x-axis. The direction of an edge at (x, y) is *perpendicular* to the direction of the gradient vector at that point.

Computation of the gradient of an image is based on obtaining the partial derivatives $\partial f/\partial x$ and $\partial f/\partial y$ at every pixel location. Let the 3×3 area shown in Fig. 10.8(a) represent the gray levels in a neighborhood of an image. As discussed in Section 3.7.3, one of the simplest ways to implement a first-order partial derivative at point z_5 is to use the following *Roberts cross-gradient operators*:

$$G_x = (z_9 - z_5) \tag{10.1-6}$$

and

$$G_y = (z_8 - z_6). \tag{10.1-7}$$

These derivatives can be implemented for an entire image by using the masks shown in Fig. 10.8(b) with the procedure discussed in Section 3.5.

Masks of size 2×2 are awkward to implement because they do not have a clear center. An approach using masks of size 3×3 is given by

$$G_x = (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3) \tag{10.1-8}$$

a
b c
d e
f g

FIGURE 10.8
A 3×3 region of an image (the z's are gray-level values) and various masks used to compute the gradient at point labeled z_5.

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

−1	0	0	−1
0	1	1	0

Roberts

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

Prewitt

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

Sobel

and

$$G_y = (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7). \tag{10.1-9}$$

In this formulation, the difference between the first and third rows of the 3×3 image region approximates the derivative in the x-direction, and the difference between the third and first columns approximates the derivative in the y-direction. The masks shown in Figs. 10.8(d) and (e), called the *Prewitt operators*, can be used to implement these two equations.

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

$$G_x = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3) \tag{10.1-10}$$

and

$$G_y = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7). \tag{10.1-11}$$

A weight value of 2 is used to achieve some smoothing by giving more importance to the center point (Problem 10.8). Figures 10.8(f) and (g), called the *Sobel* operators, are used to implement these two equations. The Prewitt and Sobel

operators are among the most used in practice for computing digital gradients. The Prewitt masks are simpler to implement than the Sobel masks, but the latter have slightly superior noise-suppression characteristics, an important issue when dealing with derivatives. Note that the coefficients in all the masks shown in Fig. 10.8 sum to 0, indicating that they give a response of 0 in areas of constant gray level, as expected of a derivative operator.

The masks just discussed are used to obtain the gradient components G_x and G_y. Computation of the gradient requires that these two components be combined in the manner shown in Eq. (10.1-4). 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 gradient by absolute values:

$$\nabla f \approx |G_x| + |G_y|. \tag{10.1-12}$$

This equation is much more attractive computationally, and it still preserves relative changes in gray levels. As discussed in Section 3.7.3, 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. These masks give isotropic results only for vertical and horizontal edges, so even if we used Eq. (10.1-4) to compute the gradient, the results would be isotropic only for edges in those directions. In this case, Eqs. (10.1-4) and (10.1-12) give the same result (Problem 10.6).

It is possible to modify the 3×3 masks in Fig. 10.8 so that they have their strongest responses along the diagonal directions. The two additional Prewitt and Sobel masks for detecting discontinuities in the diagonal directions are shown in Fig. 10.9.

EXAMPLE 10.4: Illustration of the gradient and its components.

■ Figure 10.10 illustrates the 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

0	1	1
−1	0	1
−1	−1	0

−1	−1	0
−1	0	1
0	1	1

Prewitt

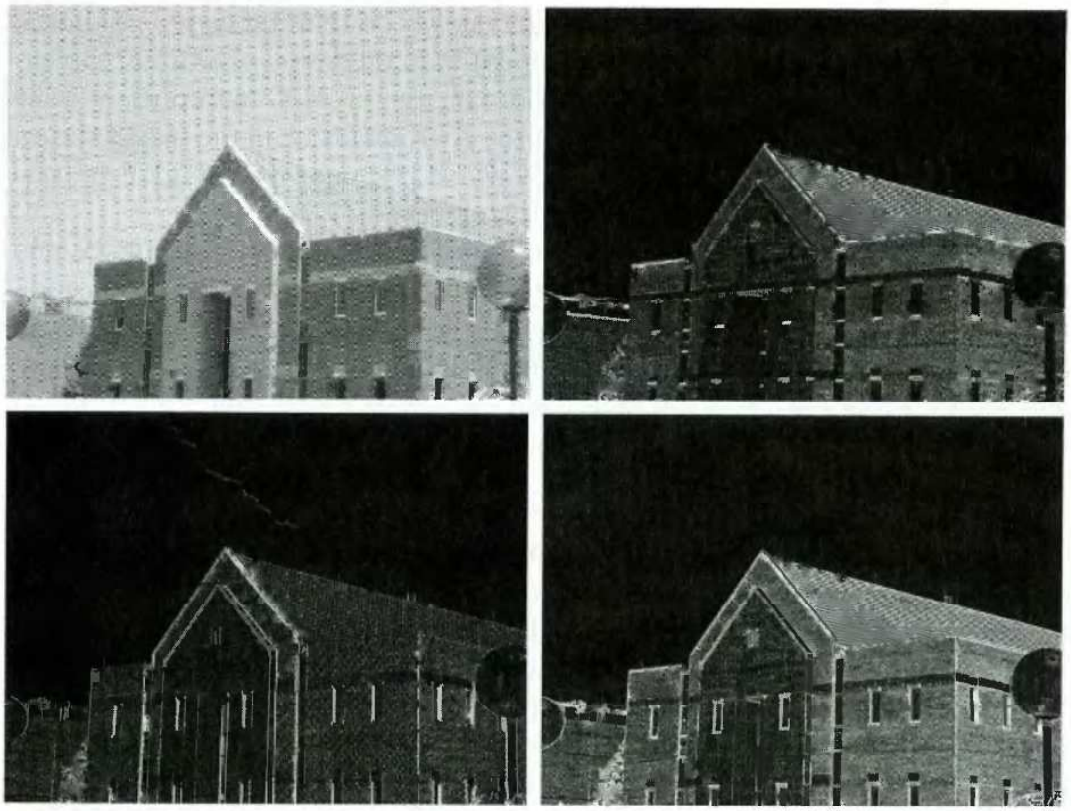
0	1	2
−1	0	1
2	1	0

−2	−1	0
−1	0	1
0	1	2

Sobel

a b
c d

FIGURE 10.9 Prewitt and Sobel masks for detecting diagonal edges.



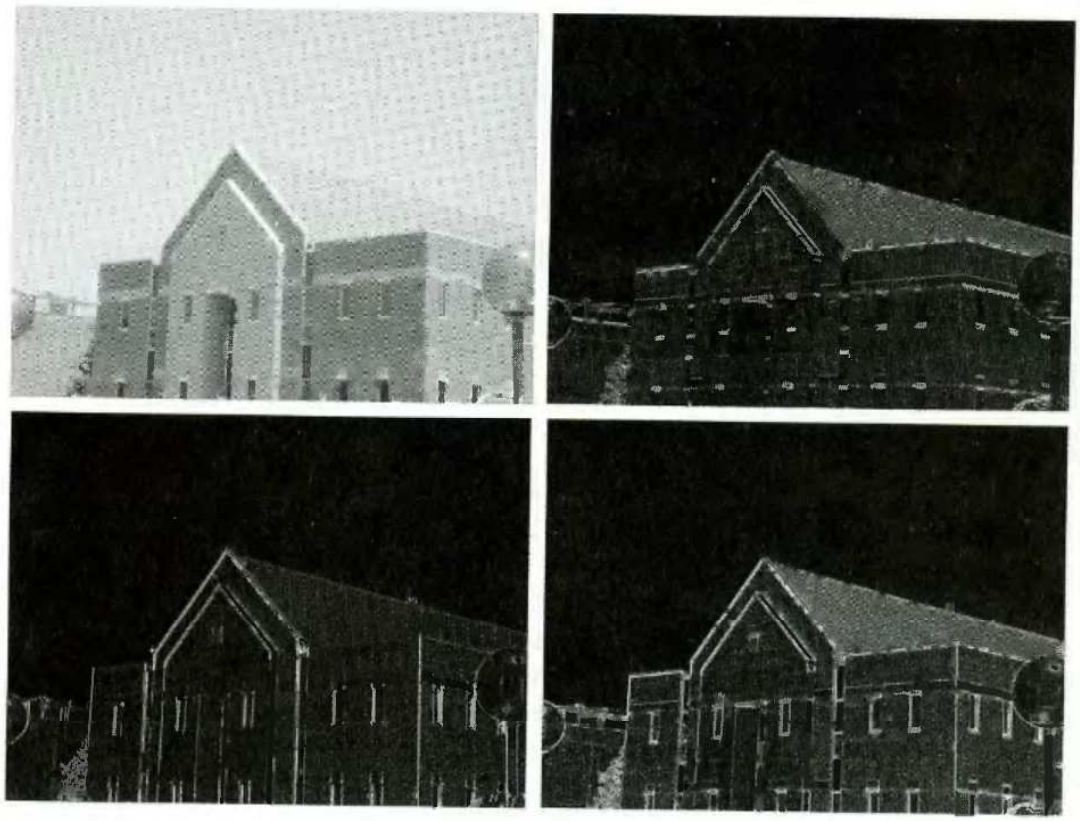
a b
c d

FIGURE 10.10 (a) Original image. (b) $|G_x|$, component of the gradient in the x-direction. (c) $|G_y|$, component in the y-direction. (d) Gradient image, $|G_x| + |G_y|$.

components. The directionality of the two components is evident in Figs. 10.10(b) and (c). Note in particular how strong the roof tile, horizontal brick joints, and horizontal segments of the windows are in Fig. 10.10(b). By contrast, Fig. 10.10(c) favors the vertical components, such as the corner of the near wall, the vertical components of the window, the vertical joints of the brick, and the lamppost on the right side of the picture.

The original image is of reasonably high resolution (1200 × 1600 pixels) and, at the distance the image was taken, the contribution made to image detail by the wall bricks is still significant. This level of detail often is undesirable, and one way to reduce it is to smooth the image. Figure 10.11 shows the same sequence of images as in Fig. 10.10, but with the original image being smoothed first using a 5 × 5 averaging filter. The response of each mask now shows almost no contribution due to the bricks, with the result being dominated mostly by the principal edges. Note that averaging caused the response of all edges to be weaker.

In Figs. 10.10 and 10.11, it is evident that the horizontal and vertical Sobel masks respond about equally well to edges oriented in the minus and plus 45° directions. If it is important to emphasize edges along the diagonal directions, then one of the mask pairs in Fig. 10.9 should be used. The absolute responses of the diagonal Sobel masks are shown in Fig. 10.12. The stronger diagonal response of these masks is evident in this figure. 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 Sobel masks shown in Figs. 10.10(b) and 10.10(c). ■



a b
c d

FIGURE 10.11 Same sequence as in Fig. 10.10, but with the original image smoothed with a 5×5 averaging filter.

a b

FIGURE 10.12 Diagonal edge detection. (a) Result of using the mask in Fig. 10.9(c). (b) Result of using the mask in Fig. 10.9(d). The input in both cases was Fig. 10.11(a).

The Laplacian

The Laplacian of a 2-D function $f(x, y)$ is a second-order derivative defined as

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}. \tag{10.1-13}$$

Digital approximations to the Laplacian were introduced in Section 3.7.2. For a 3×3 region, one of the two forms encountered most frequently in practice is

$$\nabla^2 f = 4z_5 - (z_2 + z_4 + z_6 + z_8) \tag{10.1-14}$$

FIGURE 10.13 Laplacian masks used to implement Eqs. (10.1-14) and (10.1-15), respectively.

0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

where the z's are defined in Fig. 10.8(a). A digital approximation including the diagonal neighbors is given by

$$\nabla^2 f = 8z_5 - (z_1 + z_2 + z_3 + z_4 + z_6 + z_7 + z_8 + z_9). \quad (10.1\text{-}15)$$

Masks for implementing these two equations are shown in Fig. 10.13. We note from these masks that the implementations of Eqs. (10.1-14) and (10.1-15) are isotropic for rotation increments of 90° and 45°, respectively.

The Laplacian generally is not used in its original form for edge detection for several reasons: As a second-order derivative, the Laplacian typically is unacceptably sensitive to noise (Fig. 10.7). The magnitude of the Laplacian produces double edges (see Figs. 10.6 and 10.7), an undesirable effect because it complicates segmentation. Finally, the Laplacian is unable to detect edge direction. For these reasons, the role of the Laplacian in segmentation consists of (1) using its zero-crossing property for edge location, as mentioned earlier in this section, or (2) using it for the complementary purpose of establishing whether a pixel is on the dark or light side of an edge, as we show in Section 10.3.6.

In the first category, the Laplacian is combined with smoothing as a precursor to finding edges via zero-crossings. Consider the function

$$h(r) = -e^{-\frac{r^2}{2\sigma^2}} \quad (10.1\text{-}16)$$

where $r^2 = x^2 + y^2$ and σ is the standard deviation. Convolving this function with an image blurs the image, with the degree of blurring being determined by the value of σ. The Laplacian of h (the second derivative of h with respect to r) is

$$\nabla^2 h(r) = -\left[\frac{r^2 - \sigma^2}{\sigma^4}\right] e^{-\frac{r^2}{2\sigma^2}}. \quad (10.1\text{-}17)$$

This function is commonly referred to as the *Laplacian of a Gaussian* (LoG) because Eq. (10.1-16) is in the form of a Gaussian function. Figure 10.14 shows a 3-D plot, image, and cross section of the LoG function. Also shown is a 5×5 mask that approximates $\nabla^2 h$. This approximation is not unique. Its purpose is to capture the essential *shape* of $\nabla^2 h$; that is, a positive central term, surrounded by an adjacent negative region that increases in value as a function of distance from the origin, and a zero outer region. The coefficients also must sum to zero, so that the response of the mask is zero in areas of constant gray level. A mask this small is useful only for images that are essentially noise free. Due to its shape, the Laplacian of a Gaussian sometimes is called the *Mexican hat* function.

Because the second derivative is a linear operation, convolving an image with $\nabla^2 h$ is the same as convolving the image with the Gaussian smoothing function of Eq. (10.1-16) first and then computing the Laplacian of the result.