

It is possible in some implementations of image averaging to have negative values when noise is added to an image. In fact, in the example just given, this was precisely the case because Gaussian random variables with zero mean and nonzero variance have negative as well as positive values. The images in the example were scaled using the second scaling method discussed at the end of the previous section. That is, the minimum value in a given average image was obtained and its negative was added to the image. Then all the pixels in the modified image were scaled to the range  $[0, 255]$  by multiplying each pixel in the modified image by the quantity  $255/\text{Max}$ , where Max was the maximum pixel value in that image.

### 3.5 Basics of Spatial Filtering

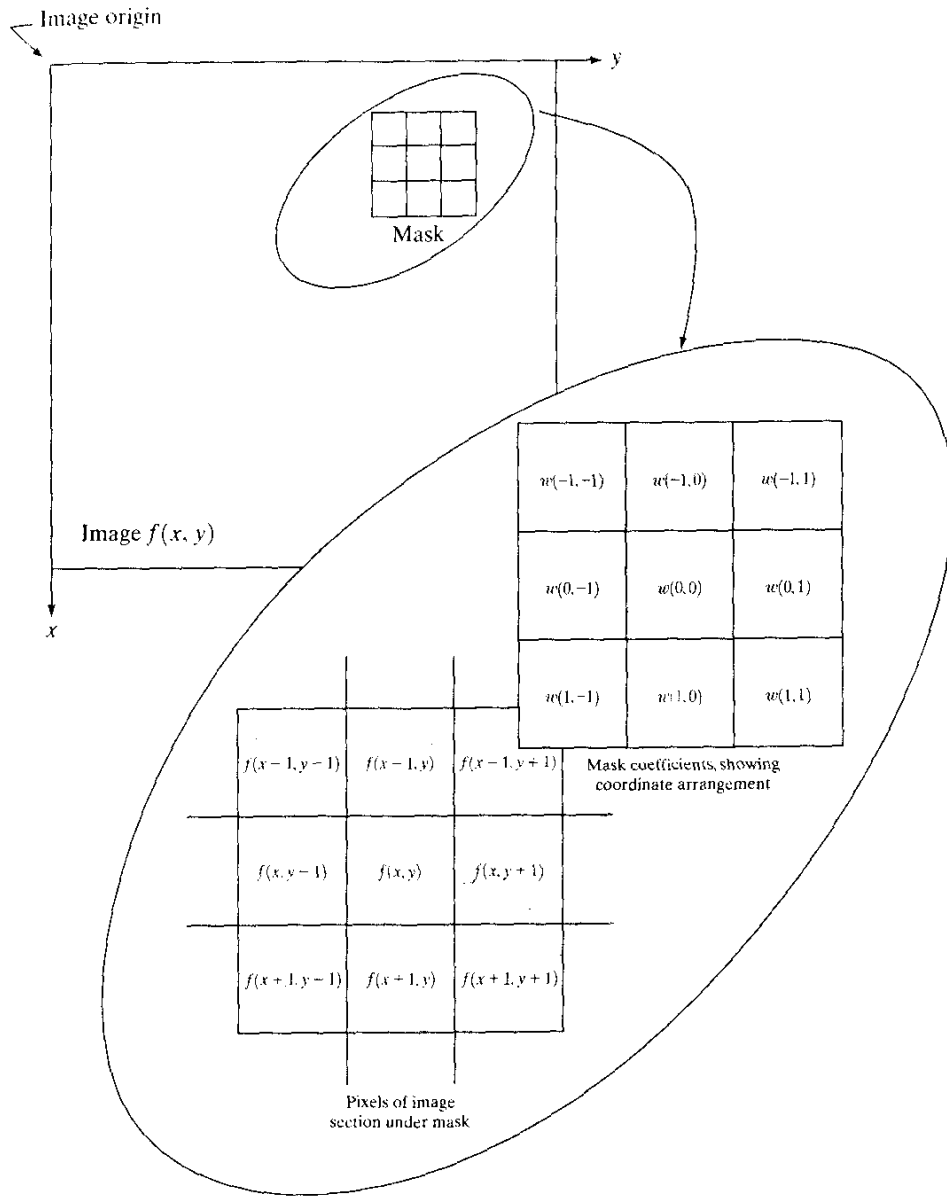
As mentioned in Section 3.1, some neighborhood operations work with the values of the image pixels in the neighborhood *and* the corresponding values of a subimage that has the same dimensions as the neighborhood. The subimage is called a *filter*, *mask*, *kernel*, *template*, or *window*, with the first three terms being the most prevalent terminology. The values in a filter subimage are referred to as *coefficients*, rather than pixels.

The concept of filtering has its roots in the use of the Fourier transform for signal processing in the so-called *frequency domain*. This topic is discussed in more detail in Chapter 4. In the present chapter, we are interested in filtering operations that are performed directly on the pixels of an image. We use the term *spatial filtering* to differentiate this type of process from the more traditional frequency domain filtering.

The mechanics of spatial filtering are illustrated in Fig. 3.32. The process consists simply of moving the filter mask from point to point in an image. At each point  $(x, y)$ , the *response* of the filter at that point is calculated using a predefined relationship. For *linear* spatial filtering (see Section 2.6 regarding linearity), the response is given by a sum of products of the filter coefficients and the corresponding image pixels in the area spanned by the filter mask. For the  $3 \times 3$  mask shown in Fig. 3.32, the result (or response),  $R$ , of linear filtering with the filter mask at a point  $(x, y)$  in the image is

$$R = w(-1, -1)f(x - 1, y - 1) + w(-1, 0)f(x - 1, y) + \cdots \\ + w(0, 0)f(x, y) + \cdots + w(1, 0)f(x + 1, y) + w(1, 1)f(x + 1, y + 1),$$

which we see is the sum of products of the mask coefficients with the corresponding pixels directly under the mask. Note in particular that the coefficient  $w(0, 0)$  coincides with image value  $f(x, y)$ , indicating that the mask is centered at  $(x, y)$  when the computation of the sum of products takes place. For a mask of size  $m \times n$ , we assume that  $m = 2a + 1$  and  $n = 2b + 1$ , where  $a$  and  $b$  are nonnegative integers. All this says is that our focus in the following discussion will be on masks of *odd* sizes, with the smallest meaningful size being  $3 \times 3$  (we exclude from our discussion the trivial case of a  $1 \times 1$  mask).



**FIGURE 3.32** The mechanics of spatial filtering. The magnified drawing shows a  $3 \times 3$  mask and the image section directly under it: the image section is shown displaced out from under the mask for ease of readability.

In general, linear filtering of an image  $f$  of size  $M \times N$  with a filter mask of size  $m \times n$  is given by the expression:

$$g(x, y) = \sum_{s=-a}^a \sum_{t=-b}^b w(s, t) f(x + s, y + t) \quad (3.5-1)$$

where, from the previous paragraph,  $a = (m - 1)/2$  and  $b = (n - 1)/2$ . To generate a complete filtered image this equation must be applied for  $x = 0, 1, 2, \dots, M - 1$  and  $y = 0, 1, 2, \dots, N - 1$ . In this way, we are assured that the

mask processes all pixels in the image. It is easily verified when  $m = n = 3$  that this expression reduces to the example given in the previous paragraph.

As discussed in Chapter 4, the process of linear filtering given in Eq. (3.5-1) is similar to a frequency domain concept called *convolution*. For this reason, linear spatial filtering often is referred to as “convolving a mask with an image.” Similarly, filter masks are sometimes called *convolution masks*. The term *convolution kernel* also is in common use.

When interest lies on the response,  $R$ , of an  $m \times n$  mask at any point  $(x, y)$ , and not on the mechanics of implementing mask convolution, it is common practice to simplify the notation by using the following expression:

$$\begin{aligned} R &= w_1 z_1 + w_2 z_2 + \dots + w_{mn} z_{mn} \\ &= \sum_{i=1}^{mn} w_i z_i \end{aligned} \quad (3.5-2)$$

where the  $w$ 's are mask coefficients, the  $z$ 's are the values of the image gray levels corresponding to those coefficients, and  $mn$  is the total number of coefficients in the mask. For the  $3 \times 3$  general mask shown in Fig. 3.33 the response at any point  $(x, y)$  in the image is given by

$$\begin{aligned} R &= w_1 z_1 + w_2 z_2 + \dots + w_9 z_9 \\ &= \sum_{i=1}^9 w_i z_i. \end{aligned} \quad (3.5-3)$$

We make special mention of this simple formula because it is seen frequently in the published literature on image processing.

Nonlinear spatial filters also operate on neighborhoods, and the mechanics of sliding a mask past an image are the same as was just outlined. In general, however, the filtering operation is based conditionally on the values of the pixels in the neighborhood under consideration, and they do not explicitly use coefficients in the sum-of-products manner described in Eqs. (3.5-1) and (3.5-2). As shown in Section 3.6.2, for example, noise reduction can be achieved effectively with a nonlinear filter whose basic function is to compute the median gray-level value in the neighborhood in which the filter is located. Computation of the median is a nonlinear operation, as is computation of the variance, which we used in Section 3.3.4.

**FIGURE 3.33**  
Another  
representation of  
a general  $3 \times 3$   
spatial filter mask.

$w_1$	$w_2$	$w_3$
$w_4$	$w_5$	$w_6$
$w_7$	$w_8$	$w_9$

An important consideration in implementing neighborhood operations for spatial filtering is the issue of what happens when the center of the filter approaches the border of the image. Consider for simplicity a square mask of size  $n \times n$ . At least one edge of such a mask will coincide with the border of the image when the center of the mask is at a distance of  $(n - 1)/2$  pixels away from the border of the image. If the center of the mask moves any closer to the border, one or more rows or columns of the mask will be located outside the image plane. There are several ways to handle this situation. The simplest is to limit the excursions of the center of the mask to be at a distance no less than  $(n - 1)/2$  pixels from the border. The resulting filtered image will be smaller than the original, but all the pixels in the filtered image will have been processed with the full mask. If the result is required to be the same size as the original, then the approach typically employed is to filter all pixels only with the section of the mask that is fully contained in the image. With this approach, there will be bands of pixels near the border that will have been processed with a partial filter mask. Other approaches include “padding” the image by adding rows and columns of 0’s (or other constant gray level), or padding by replicating rows or columns. The padding is then stripped off at the end of the process. This keeps the size of the filtered image the same as the original, but the values of the padding will have an effect near the edges that becomes more prevalent as the size of the mask increases. The only way to obtain a perfectly filtered result is to accept a somewhat smaller filtered image by limiting the excursions of the center of the filter mask to a distance no less than  $(n - 1)/2$  pixels from the border of the original image.

### Smoothing Spatial Filters

Smoothing filters are used for blurring and for noise reduction. Blurring is used in preprocessing steps, such as removal of small details from an image prior to (large) object extraction, and bridging of small gaps in lines or curves. Noise reduction can be accomplished by blurring with a linear filter and also by non-linear filtering.

#### Smoothing Linear Filters

The output (response) of a smoothing, linear spatial filter is simply the average of the pixels contained in the neighborhood of the filter mask. These filters sometimes are called *averaging filters*. For reasons explained in Chapter 4, they also are referred to as *lowpass filters*.

The idea behind smoothing filters is straightforward. By replacing the value of every pixel in an image by the average of the gray levels in the neighborhood defined by the filter mask, this process results in an image with reduced “sharp” transitions in gray levels. Because random noise typically consists of sharp transitions in gray levels, the most obvious application of smoothing is noise reduction. However, edges (which almost always are desirable features of an image) also are characterized by sharp transitions in gray levels, so averaging filters have the undesirable side effect that they blur edges. Another application of this type of process includes the smoothing of false contours that result

a b  
**FIGURE 3.34** Two  $3 \times 3$  smoothing (averaging) filter masks. The constant multiplier in front of each mask is equal to the sum of the values of its coefficients, as is required to compute an average.


from using an insufficient number of gray levels, as discussed in Section 2.4.3. A major use of averaging filters is in the reduction of “irrelevant” detail in an image. By “irrelevant” we mean pixel regions that are small with respect to the size of the filter mask. This latter application is illustrated later in this section.

Figure 3.34 shows two  $3 \times 3$  smoothing filters. Use of the first filter yields the standard average of the pixels under the mask. This can best be seen by substituting the coefficients of the mask into Eq. (3.5-3):

$$R = \frac{1}{9} \sum_{i=1}^9 z_i,$$

which is the average of the gray levels of the pixels in the  $3 \times 3$  neighborhood defined by the mask. Note that, instead of being  $1/9$ , the coefficients of the filter are all 1's. The idea here is that it is computationally more efficient to have coefficients valued 1. At the end of the filtering process the entire image is divided by 9. An  $m \times n$  mask would have a normalizing constant equal to  $1/mn$ . A spatial averaging filter in which all coefficients are equal is sometimes called a *box filter*.


The second mask shown in Fig. 3.34 is a little more interesting. This mask yields a so-called *weighted average*, terminology used to indicate that pixels are multiplied by different coefficients, thus giving more importance (weight) to some pixels at the expense of others. In the mask shown in Fig. 3.34(b) the pixel at the center of the mask is multiplied by a higher value than any other, thus giving this pixel more importance in the calculation of the average. The other pixels are inversely weighted as a function of their distance from the center of the mask. The diagonal terms are further away from the center than the orthogonal neighbors (by a factor of  $\sqrt{2}$ ) and, thus, are weighed less than these immediate neighbors of the center pixel. The basic strategy behind weighing the center point the highest and then reducing the value of the coefficients as a function of increasing distance from the origin is simply an attempt to reduce blurring in the smoothing process. We could have picked other weights to accomplish the same general objective. However, the sum of all the coefficients in the mask of Fig. 3.34(b) is equal to 16, an attractive feature for computer implementation because it has an integer power of 2. In practice, it is difficult in general to see differences between images smoothed by using either of the masks in Fig. 3.34, or similar arrangements, because the area these masks span at any one location in an image is so small.

With reference to Eq. (3.5-1), the general implementation for filtering an  $M \times N$  image with a weighted averaging filter of size  $m \times n$  ( $m$  and  $n$  odd) is given by the expression

$$g(x, y) = \frac{\sum_{s=-a}^a \sum_{t=-b}^b w(s, t) f(x + s, y + t)}{\sum_{s=-a}^a \sum_{t=-b}^b w(s, t)} \quad (3.6-1)$$

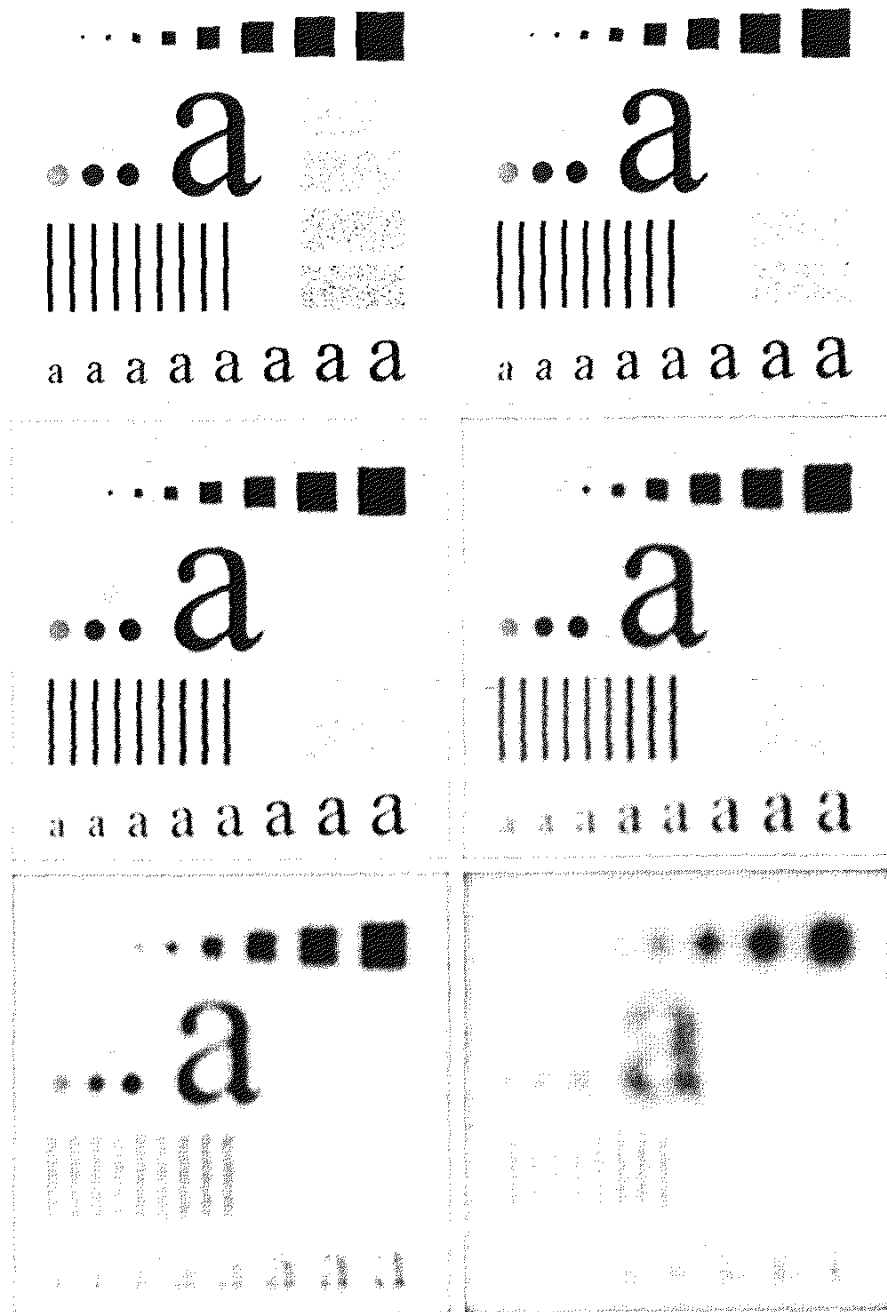
The parameters in this equation are as defined in Eq. (3.5-1). As before, it is understood that the complete filtered image is obtained by applying Eq. (3.6-1) for  $x = 0, 1, 2, \dots, M - 1$  and  $y = 0, 1, 2, \dots, N - 1$ . The denominator in Eq. (3.6-1) is simply the sum of the mask coefficients and, therefore, it is a constant that needs to be computed only once. Typically, this scale factor is applied to all the pixels of the output image after the filtering process is completed.

The effects of smoothing as a function of filter size are illustrated in Fig. 3.35, which shows an original image and the corresponding smoothed results obtained using square averaging filters of sizes  $n = 3, 5, 9, 15$ , and  $35$  pixels, respectively. The principal features of these results are as follows: For  $n = 3$ , we note a general slight blurring throughout the entire image but, as expected, details that are of approximately the same size as the filter mask are affected considerably more. For example, the  $3 \times 3$  and  $5 \times 5$  squares, the small letter "a," and the fine grain noise show significant blurring when compared to the rest of the image. A positive result is that the noise is less pronounced. Note that the jagged borders of the characters and gray circles have been pleasingly smoothed.

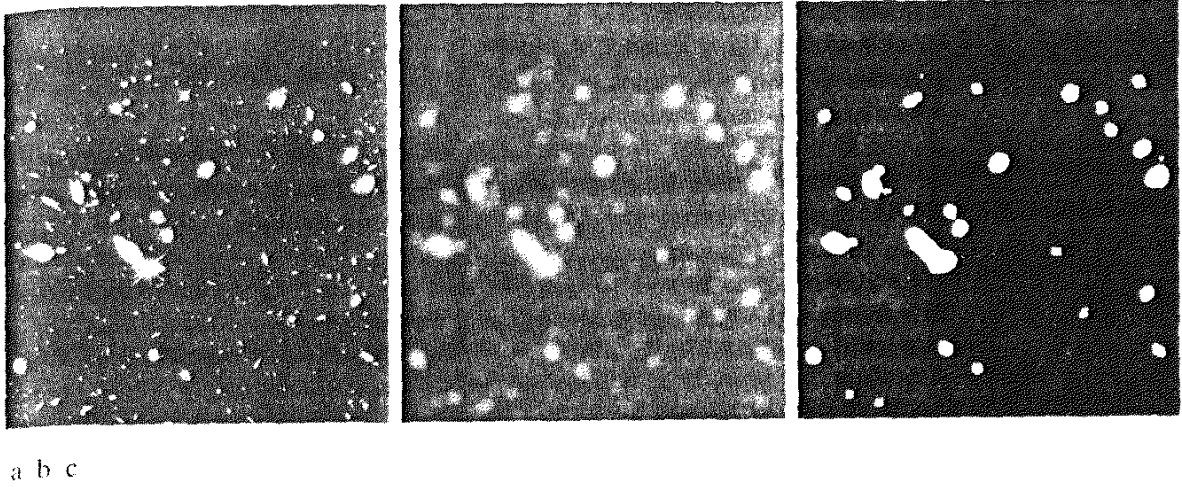
The result for  $n = 5$  is somewhat similar, with a slight further increase in blurring. For  $n = 9$  we see considerably more blurring, and the 20% black circle is not nearly as distinct from the background as in the previous three images, illustrating the blending effect that blurring has on objects whose gray level content is close to that of its neighboring pixels. Note the significant further smoothing of the noisy rectangles. The results for  $n = 15$  and  $35$  are extreme with respect to the sizes of the objects in the image. This type of excessive blurring is generally used to eliminate small objects from an image. For instance, the three small squares, two of the circles, and most of the noisy rectangle areas have been blended into the background of the image in Fig. 3.35(f). Note also in this figure the pronounced black border. This is a result of padding the border of the original image with 0's (black) and then trimming off the padded area. Some of the black was blended into all filtered images, but became truly objectionable for the images smoothed with the larger filters. 

**EXAMPLE 3.9:**  
Image smoothing  
with masks of  
various sizes.

As mentioned earlier, an important application of spatial averaging is to blur an image for the purpose getting a gross representation of objects of interest, such that the intensity of smaller objects blends with the background and larger objects become "bloblike" and easy to detect. The size of the mask establishes the relative size of the objects that will be blended with the background. As an illustration, consider Fig. 3.36(a), which is an image from the Hubble telescope in orbit around the Earth. Figure 3.36(b) shows the result of applying a



**FIGURE 3.35** (a) Original image, of size  $500 \times 500$  pixels. (b)–(f) Results of smoothing with square averaging filter masks of sizes  $n = 3, 5, 9, 15$ , and  $35$ , respectively. The black squares at the top are of sizes  $3, 5, 9, 15, 25, 35, 45$ , and  $55$  pixels, respectively; their borders are  $25$  pixels apart. The letters at the bottom range in size from  $10$  to  $24$  points, in increments of  $2$  points; the large letter at the top is  $60$  points. The vertical bars are  $5$  pixels wide and  $100$  pixels high; their separation is  $20$  pixels. The diameter of the circles is  $25$  pixels, and their borders are  $15$  pixels apart; their gray levels range from  $0\%$  to  $100\%$  black in increments of  $20\%$ . The background of the image is  $10\%$  black. The noisy rectangles are of size  $50 \times 120$  pixels.



**FIGURE 3.36** (a) Image from the Hubble Space Telescope. (b) Image processed by a  $15 \times 15$  averaging mask. (c) Result of thresholding (b). (Original image courtesy of NASA.)

$15 \times 15$  averaging mask to this image. We see that a number of objects have either blended with the background or their intensity has diminished considerably. It is typical to follow an operation like this with thresholding to eliminate objects based on their intensity. The result of using the thresholding function of Fig. 3.2(b) with a threshold value equal to 25% of the highest intensity in the blurred image is shown in Fig. 3.36(c). Comparing this result with the original image, we see that it is a reasonable representation of what we would consider to be the largest, brightest objects in that image.

### Order-Statistics Filters

Order-statistics filters are nonlinear spatial filters whose response is based on ordering (ranking) the pixels contained in the image area encompassed by the filter, and then replacing the value of the center pixel with the value determined by the ranking result. The best-known example in this category is the *median filter*, which, as its name implies, replaces the value of a pixel by the median of the gray levels in the neighborhood of that pixel (the original value of the pixel is included in the computation of the median). Median filters are quite popular because, for certain types of random noise, they provide excellent noise-reduction capabilities, with considerably less blurring than linear smoothing filters of similar size. Median filters are particularly effective in the presence of *impulse noise*, also called *salt-and-pepper noise* because of its appearance as white and black dots superimposed on an image.

The median,  $\xi$ , of a set of values is such that half the values in the set are less than or equal to  $\xi$ , and half are greater than or equal to  $\xi$ . In order to perform median filtering at a point in an image, we first sort the values of the pixel in question and its neighbors, determine their median, and assign this value to that pixel. For example, in a  $3 \times 3$  neighborhood the median is the 5th largest value, in a  $5 \times 5$  neighborhood the 13th largest value, and so on. When several values

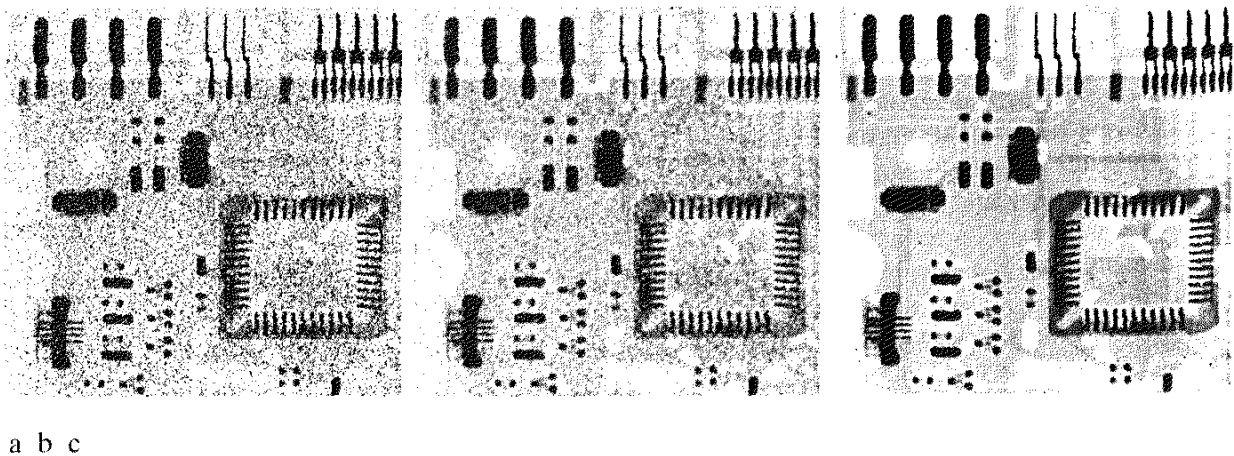


in a neighborhood are the same, all equal values are grouped. For example, suppose that a  $3 \times 3$  neighborhood has values (10, 20, 20, 20, 15, 20, 20, 25, 100). These values are sorted as (10, 15, 20, 20, 20, 20, 20, 25, 100), which results in a median of 20. Thus, the principal function of median filters is to force points with distinct gray levels to be more like their neighbors. In fact, isolated clusters of pixels that are light or dark with respect to their neighbors, and whose area is less than  $n^2/2$  (one-half the filter area), are eliminated by an  $n \times n$  median filter. In this case “eliminated” means forced to the median intensity of the neighbors. Larger clusters are affected considerably less.

Although the median filter is by far the most useful order-statistics filter in image processing, it is by no means the only one. The median represents the 50th percentile of a ranked set of numbers, but the reader will recall from basic statistics that ranking lends itself to many other possibilities. For example, using the 100th percentile results in the so-called *max filter*, which is useful in finding the brightest points in an image. The response of a  $3 \times 3$  max filter is given by  $R = \max\{z_k | k = 1, 2, \dots, 9\}$ . The 0th percentile filter is the *min filter*, used for the opposite purpose. Median, max, and mean filters are considered in more detail in Chapter 5.

**EXAMPLE 3.10:**  
Use of median  
filtering for noise  
reduction.

■ Figure 3.37(a) shows an X-ray image of a circuit board heavily corrupted by salt-and-pepper noise. To illustrate the point about the superiority of median filtering over average filtering in situations such as this, we show in Fig. 3.37(b) the result of processing the noisy image with a  $3 \times 3$  neighborhood averaging mask, and in Fig. 3.37(c) the result of using a  $3 \times 3$  median filter. The image processed with the averaging filter has less visible noise, but the price paid is significant blurring. The superiority in all respects of median over average filtering in this case is quite evident. In general, median filtering is much better suited than averaging for the removal of additive salt-and-pepper noise.



**FIGURE 3.37** (a) X-ray image of circuit board corrupted by salt-and-pepper noise. (b) Noise reduction with a  $3 \times 3$  averaging mask. (c) Noise reduction with a  $3 \times 3$  median filter. (Original image courtesy of Mr. Joseph E. Pascente, Lixi, Inc.)

## Sharpening Spatial Filters

The principal objective of sharpening is to highlight fine detail in an image or to enhance detail that has been blurred, either in error or as a natural effect of a particular method of image acquisition. Uses of image sharpening vary and include applications ranging from electronic printing and medical imaging to industrial inspection and autonomous guidance in military systems.

In the last section, we saw that image blurring could be accomplished in the spatial domain by pixel averaging in a neighborhood. Since averaging is analogous to integration, it is logical to conclude that sharpening could be accomplished by spatial differentiation. This, in fact, is the case, and the discussion in this section deals with various ways of defining and implementing operators for sharpening by digital differentiation. Fundamentally, the strength of the response of a derivative operator is proportional to the degree of discontinuity of the image at the point at which the operator is applied. Thus, image differentiation enhances edges and other discontinuities (such as noise) and deemphasizes areas with slowly varying gray-level values.

### 3.7.1 Foundation

In the two sections that follow, we consider in some detail sharpening filters that are based on first- and second-order derivatives, respectively. Before proceeding with that discussion, however, we stop to look at some of the fundamental properties of these derivatives in a digital context. To simplify the explanation, we focus attention on one-dimensional derivatives. In particular, we are interested in the behavior of these derivatives in areas of constant gray level (flat segments), at the onset and end of discontinuities (step and ramp discontinuities), and along gray-level ramps. These types of discontinuities can be used to model noise points, lines, and edges in an image. The behavior of derivatives during transitions into and out of these image features also is of interest.

The derivatives of a digital function are defined in terms of differences. There are various ways to define these differences. However, we require that any definition we use for a first derivative (1) must be zero in flat segments (areas of constant gray-level values); (2) must be nonzero at the onset of a gray-level step or ramp; and (3) must be nonzero along ramps. Similarly, any definition of a second derivative (1) must be zero in flat areas; (2) must be nonzero at the onset and end of a gray-level step or ramp; and (3) must be zero along ramps of constant slope. Since we are dealing with digital quantities whose values are finite, the maximum possible gray-level change also is finite, and the shortest distance over which that change can occur is between adjacent pixels.

A basic definition of the first-order derivative of a one-dimensional function  $f(x)$  is the difference

$$\frac{\partial f}{\partial x} = f(x + 1) - f(x).$$

We used a partial derivative here in order to keep the notation the same as when we consider an image function of two variables,  $f(x, y)$ , at which time we

will be dealing with partial derivatives along the two spatial axes. Use of a partial derivative in the present discussion does not affect in any way the nature of what we are trying to accomplish.

Similarly, we define a second-order derivative as the difference

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

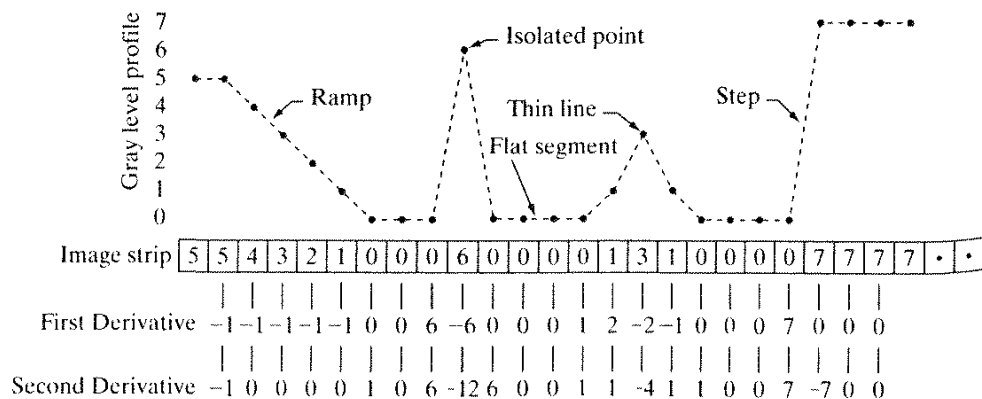
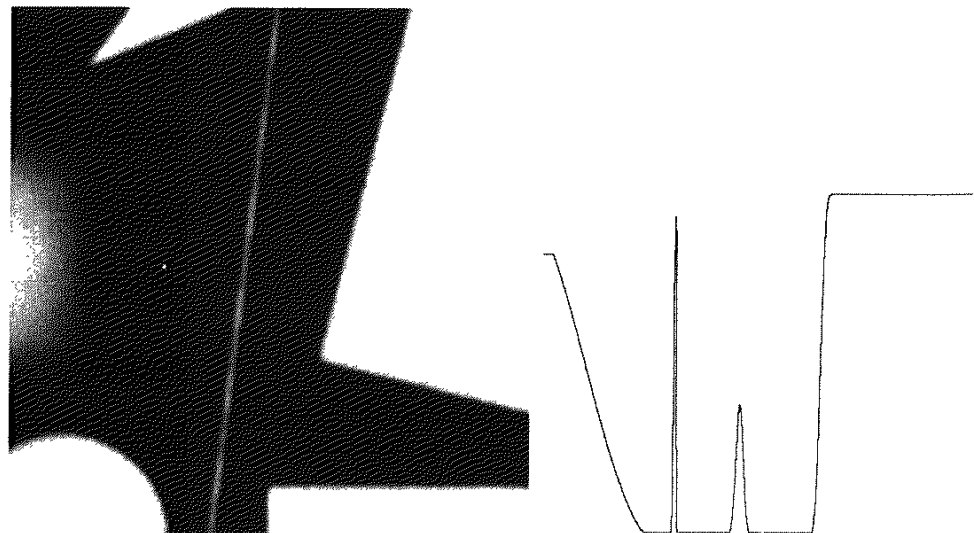
It is easily verified that these two definitions satisfy the conditions stated previously regarding derivatives of the first and second order. To see this, and also to highlight the fundamental similarities and differences between first- and second-order derivatives in the context of image processing, consider the example shown in Fig. 3.38.

Figure 3.38(a) shows a simple image that contains various solid objects, a line, and a single noise point. Figure 3.38(b) shows a horizontal gray-level profile (scan line) of the image along the center and including the isolated noise point. This profile is the one-dimensional function we will use for illustrations regarding this figure. Figure 3.38(c) shows a simplification of the profile, with just enough num-

a b  
c

**FIGURE 3.38**

(a) A simple image. (b) 1-D horizontal gray-level profile along the center of the image and including the isolated noise point. (c) Simplified profile (the points are joined by dashed lines to simplify interpretation).



bers to make it possible for us to analyze how the first- and second-order derivatives behave as they encounter a noise point, a line, and then the edge of an object. In our simplified diagram the transition in the ramp spans four pixels, the noise point is a single pixel, the line is three pixels thick, and the transition into the gray-level step takes place between adjacent pixels. The number of gray levels was simplified to only eight levels.

Let us consider the properties of the first and second derivatives as we traverse the profile from left to right. First, we note that the first-order derivative is nonzero along the entire ramp, while the second-order derivative is *nonzero only* at the onset and end of the ramp. Because edges in an image resemble this type of transition, we conclude that first-order derivatives produce “thick” edges and second-order derivatives, much *finer* ones. Next we encounter the isolated noise point. Here, the response at and around the point is much stronger for the second- than for the first-order derivative. Of course, this is not unexpected. A second-order derivative is much more aggressive than a first-order derivative in enhancing sharp changes. Thus, we can expect a second-order derivative to enhance fine detail (including noise) much more than a first-order derivative. The thin line is a fine detail, and we see essentially the same difference between the two derivatives. If the maximum gray level of the line had been the same as the isolated point, the response of the second derivative would have been stronger for the latter. Finally, in this case, the response of the two derivatives is the same at the gray-level step (in most cases when the transition into a step is not from zero, the second derivative will be weaker). We also note that the second derivative has a transition from positive back to negative. In an image, this shows as a thin double line. This “double-edge” effect is an issue that will be important in Chapter 10, where we use derivatives for edge detection. It is of interest also to note that if the gray level of the thin line had been the same as the step, the response of the second derivative would have been stronger for the line than for the step.

In summary, comparing the response between first- and second-order derivatives, we arrive at the following conclusions. (1) First-order derivatives generally produce thicker edges in an image. (2) Second-order derivatives have a stronger response to fine detail, such as thin lines and isolated points. (3) First-order derivatives generally have a stronger response to a gray-level step. (4) Second-order derivatives produce a double response at step changes in gray level. We also note of second-order derivatives that, for similar changes in gray-level values in an image, their response is stronger to a line than to a step, and to a point than to a line.

In most applications, the second derivative is better suited than the first derivative for image enhancement because of the ability of the former to enhance fine detail. For this, and for reasons of simpler implementation and extensions, we will focus attention initially on uses of the second derivative for enhancement. First-order derivatives are discussed in Section 3.7.3. Although the principle of use of first derivatives in image processing is for edge extraction, they do have important uses in image enhancement. In fact, we show in Section 3.8 that they can be used in conjunction with the second derivative to obtain some impressive enhancement results.

### 3.7.2 Use of Second Derivatives for Enhancement—The Laplacian

In this section we consider in some detail the use of two-dimensional, second-order derivatives for image enhancement. The approach basically consists of defining a discrete formulation of the second-order derivative and then constructing a filter mask based on that formulation. We are interested in *isotropic* filters, whose response is independent of the direction of the discontinuities in the image to which the filter is applied. In other words, isotropic filters are *rotation invariant*, in the sense that rotating the image and then applying the filter gives the same result as applying the filter to the image first and then rotating the result.

#### Development of the method

It can be shown (Rosenfeld and Kak [1982]) that the simplest isotropic derivative operator is the *Laplacian*, which, for a function (image)  $f(x, y)$  of two variables, is defined as

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}. \quad (3.7-1)$$

Because derivatives of any order are linear operations, the Laplacian is a linear operator.

In order to be useful for digital image processing, this equation needs to be expressed in discrete form. There are several ways to define a digital Laplacian using neighborhoods. Whatever the definition, however, it has to satisfy the properties of a second derivative outlined in Section 3.7.1. The definition of the digital second derivative given in that section is one of the most used. Taking into account that we now have two variables, we use the following notation for the partial second-order derivative in the  $x$ -direction:

$$\frac{\partial^2 f}{\partial^2 x^2} = f(x + 1, y) + f(x - 1, y) - 2f(x, y) \quad (3.7-2)$$

and, similarly in the  $y$ -direction, as

$$\frac{\partial^2 f}{\partial^2 y^2} = f(x, y + 1) + f(x, y - 1) - 2f(x, y) \quad (3.7-3)$$

The digital implementation of the two-dimensional Laplacian in Eq. (3.7-1) is obtained by summing these two components:

$$\begin{aligned} \nabla^2 f = & [f(x + 1, y) + f(x - 1, y) + f(x, y + 1) + f(x, y - 1)] \\ & - 4f(x, y). \end{aligned} \quad (3.7-4)$$

This equation can be implemented using the mask shown in Fig. 3.39(a), which gives an isotropic result for rotations in increments of  $90^\circ$ . The mechanics of implementation are given in Eq. (3.5-1) and are illustrated in Section 3.6.1 for the linear smoothing filters. We simply are using different coefficients here.

The diagonal directions can be incorporated in the definition of the digital Laplacian by adding two more terms to Eq. (3.7-4), one for each of the two diagonal directions. The form of each new term is the same as either Eq. (3.7-2)

0	1	0	1	1	1
1	-4	1	1	-8	1
0	1	0	1	1	1

0	-1	0	-1	-1	-1
-1	4	-1	-1	8	-1
0	-1	0	-1	-1	-1

a b  
c d

**FIGURE 3.39**

(a) Filter mask used to implement the digital Laplacian, as defined in Eq. (3.7-4). (b) Mask used to implement an extension of this equation that includes the diagonal neighbors. (c) and (d) Two other implementations of the Laplacian

or (3.7-3), but the coordinates are along the diagonals. Since each diagonal term also contains a  $-2f(x, y)$  term, the total subtracted from the difference terms now would be  $-8f(x, y)$ . The mask used to implement this new definition is shown in Fig. 3.39(b). This mask yields isotropic results for increments of  $45^\circ$ . The other two masks shown in Fig. 3.39 also are used frequently in practice. They are based on a definition of the Laplacian that is the negative of the one we used here. As such, they yield equivalent results, but the difference in sign must be kept in mind when combining (by addition or subtraction) a Laplacian-filtered image with another image.

Because the Laplacian is a derivative operator, its use highlights gray-level discontinuities in an image and deemphasizes regions with slowly varying gray levels. This will tend to produce images that have grayish edge lines and other discontinuities, all superimposed on a dark, featureless background. Background features can be “recovered” while still preserving the sharpening effect of the Laplacian operation simply by adding the original and Laplacian images. As noted in the previous paragraph, it is important to keep in mind which definition of the Laplacian is used. If the definition used has a negative center coefficient, then we *subtract*, rather than add, the Laplacian image to obtain a sharpened result. Thus, the basic way in which we use the Laplacian for image enhancement is as follows:

$$g(x, y) = \begin{cases} f(x, y) - \nabla^2 f(x, y) & \text{if the center coefficient of the} \\ & \text{Laplacian mask is negative} \\ f(x, y) + \nabla^2 f(x, y) & \text{if the center coefficient of the} \\ & \text{Laplacian mask is positive.} \end{cases} \quad (3.7-5)$$

Use of this equation is illustrated next.

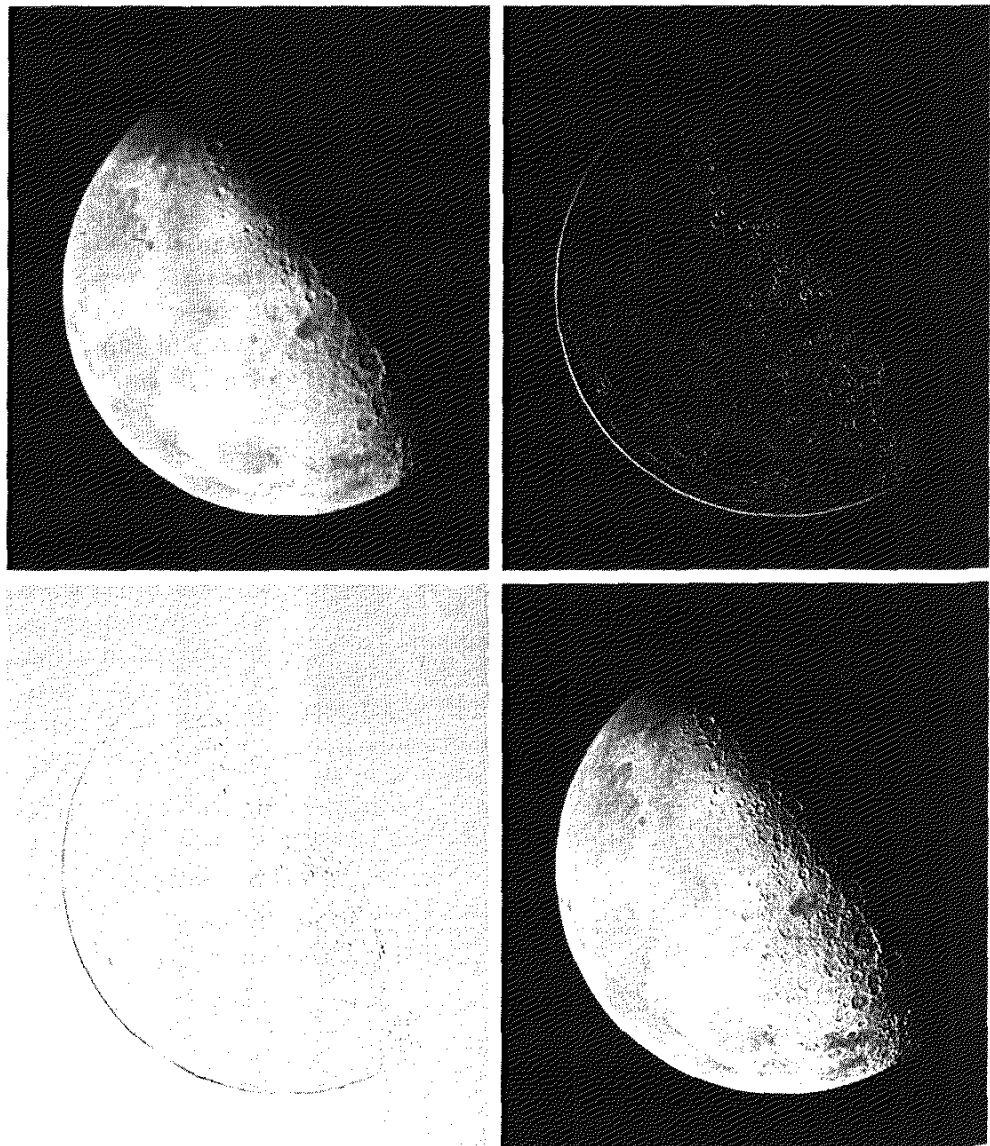
**EXAMPLE 3.11:**  
Imaging  
sharpening with  
the Laplacian.

Figure 3.40(a) shows an image of the North Pole of the moon. Figure 3.40(b) shows the result of filtering this image with the Laplacian mask in Fig. 3.39(b). Since the Laplacian image contains both positive and negative values, a typical way to scale it is to use the approach discussed at the end of Section 3.4.1. Sometimes one encounters the absolute value being used for this purpose, but this really is not correct because it produces double lines of nearly equal magnitude, which can be confusing.

The image shown in Fig. 3.40(c) was scaled in the manner just described for display purposes. Note that the dominant features of the image are edges and sharp gray-level discontinuities of various gray-level values. The background, previously near black, is now gray due to the scaling. This grayish appearance is typical of Laplacian images that have been scaled properly. Finally, Fig. 3.40(d)

a b  
c d

**FIGURE 3.40**  
(a) Image of the North Pole of the moon.  
(b) Laplacian-filtered image.  
(c) Laplacian image scaled for display purposes.  
(d) Image enhanced by using Eq. (3.7-5).  
(Original image courtesy of NASA.)



shows the result obtained using Eq. (3.7-5). The detail in this image is unmistakably clearer and sharper than in the original image. Adding the image to the Laplacian restored the overall gray level variations in the image, with the Laplacian increasing the contrast at the locations of gray-level discontinuities. The net result is an image in which small details were enhanced and the background tonality was perfectly preserved. Results like these have made Laplacian-based enhancement a fundamental tool used frequently for sharpening digital images.

### Simplifications

In the previous example, we implemented Eq. (3.7-5) by first computing the Laplacian-filtered image and then subtracting it from the original image. This was done for instructional purposes to illustrate each step in the procedure. In practice, Eq. (3.7-5) is usually implemented with one pass of a single mask. The coefficients of the single mask are easily obtained by substituting Eq. (3.7-4) for  $\nabla^2 f(x, y)$  in the first line of Eq. (3.7-5):

$$\begin{aligned} g(x, y) &= f(x, y) - [f(x+1, y) + f(x-1, y) \\ &\quad + f(x, y+1) + f(x, y-1)] + 4f(x, y) \\ &= 5f(x, y) - [f(x+1, y) + f(x-1, y) \\ &\quad + f(x, y+1) + f(x, y-1)]. \end{aligned} \quad (3.7-6)$$

This equation can be implemented using the mask shown in Fig. 3.41(a). The mask shown in Fig. 3.41(b) would be used if the diagonal neighbors also were included in the calculation of the Laplacian. Identical masks would have resulted if we had substituted the negative of Eq. (3.7-4) into the second line of Eq. (3.7-5).

The results obtainable with the mask containing the diagonal terms usually are a little sharper than those obtained with the more basic mask of Fig. 3.41(a). This property is illustrated by the Laplacian-filtered images shown in Figs. 3.41(d) and (e), which were obtained by using the masks in Figs. 3.41(a) and (b), respectively. By comparing the filtered images with the original image shown in Fig. 3.41(c), we note that both masks produced effective enhancement, but the result using the mask in Fig. 3.41(b) is visibly sharper. Figure 3.41(c) is a scanning electron microscope (SEM) image of a tungsten filament following thermal failure; the magnification is approximately 250 $\times$ .)

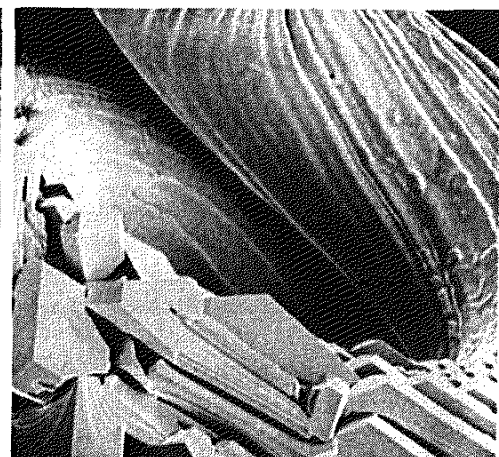
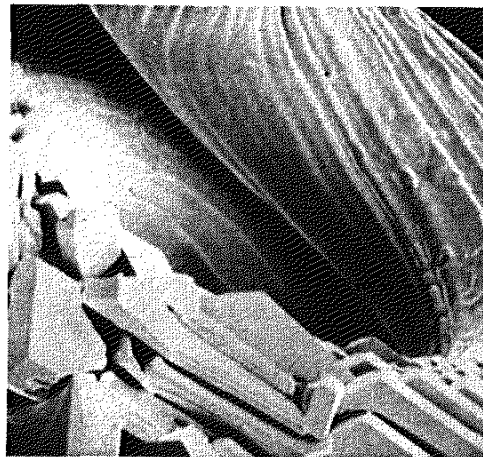
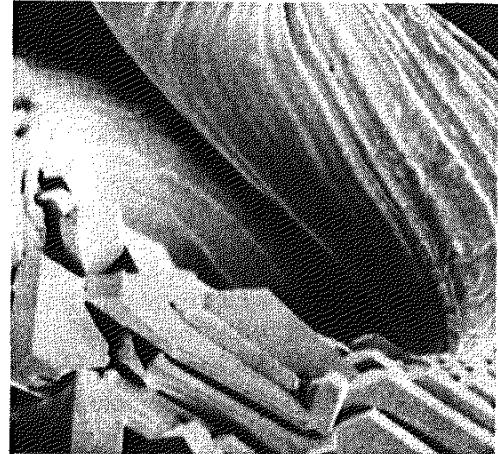
Because the Laplacian is a linear operator, we could have arrived at the same composite masks in Figs. 3.41(a) and (b) by noting that Eq. (3.7-5) is the difference between (sum of) two linear processes. That is,  $f(x, y)$  be may viewed as itself processed with a mask that has a unit coefficient in the center and zeros elsewhere. The second term in the equation is the same image processed with one of the Laplacian masks of Fig. 3.39. Due to linearity, the result obtained in Eq. (3.7-5) with the unit-center mask and one of those Laplacian masks would be the same as the result obtained with a single mask formed by subtracting (adding) the Laplacian mask from (to) the unity-center mask.

**EXAMPLE 3.12:**  
Image enhancement using a composite Laplacian mask.



0	-1	0
-1	5	-1
0	-1	0

-1	-1	-1
-1	9	-1
-1	-1	-1



**FIGURE 3.41** (a) Composite Laplacian mask. (b) A second composite mask. (c) Scanning electron microscope image. (d) and (e) Results of filtering with the masks in (a) and (b), respectively. Note how much sharper (e) is than (d). (Original image courtesy of Mr. Michael Shaffer, Department of Geological Sciences, University of Oregon, Eugene.)

### Unsharp masking and high-boost filtering

A process used for many years in the publishing industry to sharpen images consists of subtracting a blurred version of an image from the image itself. This process, called *unsharp masking*, is expressed as

$$f_s(x, y) = f(x, y) - \bar{f}(x, y) \quad (3.7-7)$$

where  $f_s(x, y)$  denotes the sharpened image obtained by unsharp masking, and  $\bar{f}(x, y)$  is a blurred version of  $f(x, y)$ . The origin of unsharp masking is in dark-room photography, where it consists of clamping together a blurred negative to a corresponding positive film and then developing this combination to produce a sharper image.

A slight further generalization of unsharp masking is called *high-boost filtering*. A high-boost filtered image,  $f_{hb}$ , is defined at any point  $(x, y)$  as

$$f_{hb}(x, y) = Af(x, y) - \bar{f}(x, y) \quad (3.7-8)$$

0	-1	0	-1	-1	-1
-1	$A + 4$	-1	-1	$A + 8$	-1
0	-1	0	-1	-1	-1

a b

**FIGURE 3.42** The high-boost filtering technique can be implemented with either one of these masks, with  $A \geq 1$ .

where  $A \geq 1$  and, as before,  $\bar{f}$  is a blurred version of  $f$ . This equation may be written as

$$f_{\text{hb}}(x, y) = (A - 1)f(x, y) + f(x, y) - \bar{f}(x, y). \quad (3.7-9)$$

By using Eq. (3.7-7), we obtain

$$f_{\text{hb}}(x, y) = (A - 1)f(x, y) + f_s(x, y) \quad (3.7-10)$$

as the expression for computing a high-boost-filtered image.

Equation (3.7-10) is applicable in general and does not state explicitly how the sharp image is obtained. If we elect to use the Laplacian, then we know that  $f_s(x, y)$  can be obtained using Eq. (3.7-5). In this case, Eq. (3.7-10) becomes

$$f_{\text{hb}} = \begin{cases} Af(x, y) - \nabla^2 f(x, y) & \text{if the center coefficient of the} \\ & \text{Laplacian mask is negative} \\ Af(x, y) + \nabla^2 f(x, y) & \text{if the center coefficient of the} \\ & \text{Laplacian mask is positive.} \end{cases} \quad (3.7-11)$$

High-boost filtering can be implemented with one pass using either of the two masks shown in Fig. 3.42. Note that, when  $A = 1$ , high-boost filtering becomes “standard” Laplacian sharpening. As the value of  $A$  increases past 1, the contribution of the sharpening process becomes less and less important. Eventually, if  $A$  is large enough, the high-boost image will be approximately equal to the original image multiplied by a constant.

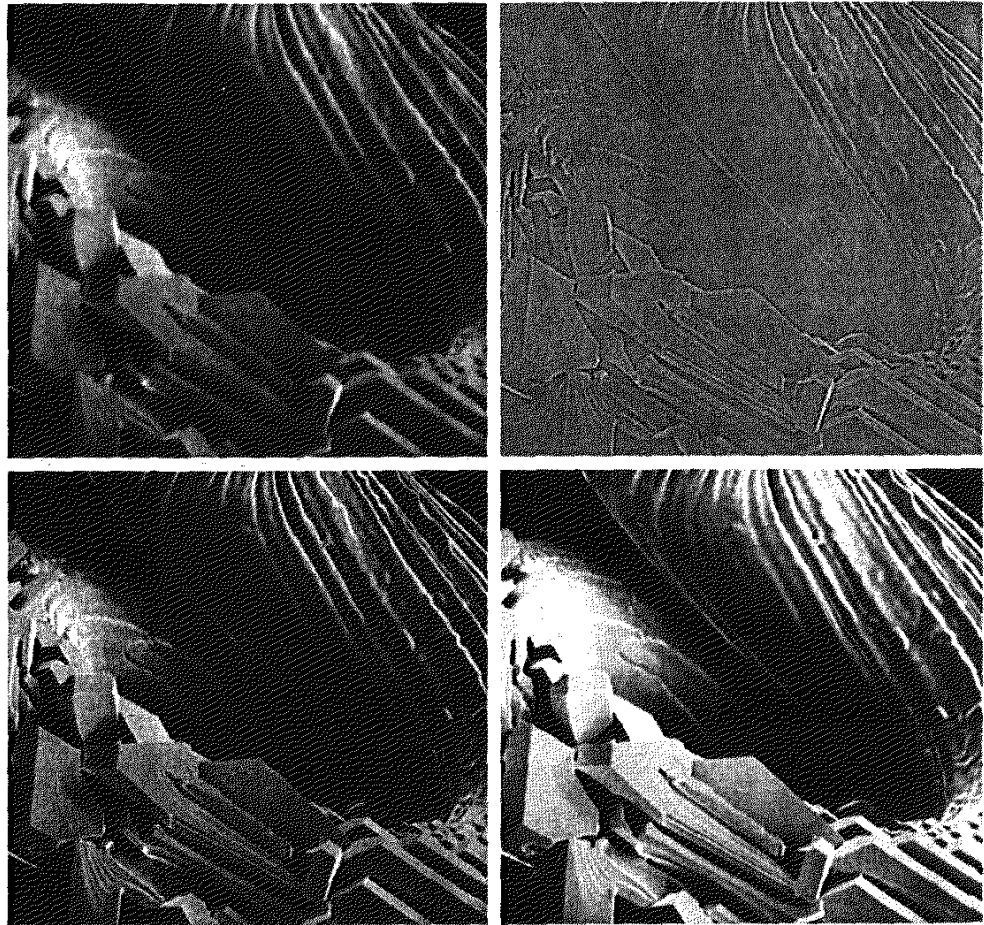
One of the principal applications of boost filtering is when the input image is darker than desired. By varying the boost coefficient, it generally is possible to obtain an overall increase in average gray level of the image, thus helping to brighten the final result. Figure 3.43 shows such an application. Part (a) of this figure is a darker version of the image in Fig. 3.41(c). Figure 3.43(b) shows the Laplacian computed using the mask in Fig. 3.42(b), with  $A = 0$ . Figure 3.43(c) was obtained using the mask in Fig. 3.42(b) with  $A = 1$ . As expected, the image has been sharpened, but it is still as dark as the original. Finally, Fig. 3.43(d) shows the result of using  $A = 1.7$ . This is a much more acceptable result, in which the average gray level has increased, thus making the image lighter and more natural.

**EXAMPLE 3.13:**  
Image enhancement with a high-boost filter.

a b  
c d

**FIGURE 3.43**

(a) Same as Fig. 3.41(c), but darker.  
(a) Laplacian of (a) computed with the mask in Fig. 3.42(b) using  $A = 0$ .  
(c) Laplacian enhanced image using the mask in Fig. 3.42(b) with  $A = 1$ . (d) Same as (c), but using  $A = 1.7$



### 3.7.3 Use of First Derivatives for Enhancement—The Gradient

First derivatives in image processing are implemented using the magnitude of the gradient. For a function  $f(x, y)$ , the gradient of  $f$  at coordinates  $(x, y)$  is defined as the two-dimensional column vector

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \quad (3.7-12)$$

The magnitude of this vector is given by

$$\begin{aligned} \nabla f &= \text{mag}(\nabla f) \\ &= [G_x^2 + G_y^2]^{1/2} \\ &= \left[ \left( \frac{\partial f}{\partial x} \right)^2 + \left( \frac{\partial f}{\partial y} \right)^2 \right]^{1/2} \end{aligned} \quad (3.7-13)$$

The components of the gradient vector itself are linear operators, but the magnitude of this vector obviously is not because of the squaring and square root

operations. On the other hand, the partial derivatives in Eq. (3.7-12) are not rotation invariant (isotropic), but the magnitude of the gradient vector is. Although it is not strictly correct, the magnitude of the gradient vector often is referred to as the *gradient*. In keeping with tradition, we will use this term in the following discussions, explicitly referring to the vector or its magnitude only in cases where confusion is likely.

The computational burden of implementing Eq. (3.7-13) over an entire image is not trivial, and it is common practice to approximate the magnitude of the gradient by using absolute values instead of squares and square roots:

$$\nabla f \approx |G_x| + |G_y|. \quad (3.7-14)$$

This equation is simpler to compute and it still preserves relative changes in gray levels, but the isotropic feature property is lost in general. However, as in the case of the Laplacian, the isotropic properties of the digital gradient defined in the following paragraph are preserved only for a limited number of rotational increments that depend on the masks used to approximate the derivatives. As it turns out, the most popular masks used to approximate the gradient give the same result only for vertical and horizontal edges and thus the isotropic properties of the gradient are preserved only for multiples of  $90^\circ$ . These results are independent of whether Eq. (3.7-13) or (3.7-14) is used, so nothing of significance is lost in using the simpler of the two equations.

As in the case of the Laplacian, we now define digital approximations to the preceding equations, and from there formulate the appropriate filter masks. In order to simplify the discussion that follows, we will use the notation in Fig. 3.44(a) to denote image points in a  $3 \times 3$  region. For example, the center point,  $z_5$ , denotes  $f(x, y)$ ,  $z_1$  denotes  $f(x - 1, y - 1)$ , and so on. As indicated in Section 3.7.1, the simplest approximations to a first-order derivative that satisfy the conditions stated in that section are  $G_x = (z_8 - z_5)$  and  $G_y = (z_6 - z_5)$ . Two other definitions proposed by Roberts [1965] in the early development of digital image processing use cross differences:

$$G_x = (z_9 - z_5) \quad \text{and} \quad G_y = (z_8 - z_6). \quad (3.7-15)$$

If we elect to use Eq. (3.7-13), then we compute the gradient as

$$\nabla f = [(z_9 - z_5)^2 + (z_8 - z_6)^2]^{1/2} \quad (3.7-16)$$

If we use absolute values, then substituting the quantities in Eq. (3.7-15) into Eq. (3.7-14) gives us the following approximation to the gradient:

$$\nabla f \approx |z_9 - z_5| + |z_8 - z_6|. \quad (3.7-17)$$

This equation can be implemented with the two masks shown in Figs. 3.44(b) and (c). These masks are referred to as the *Roberts cross-gradient operators*.

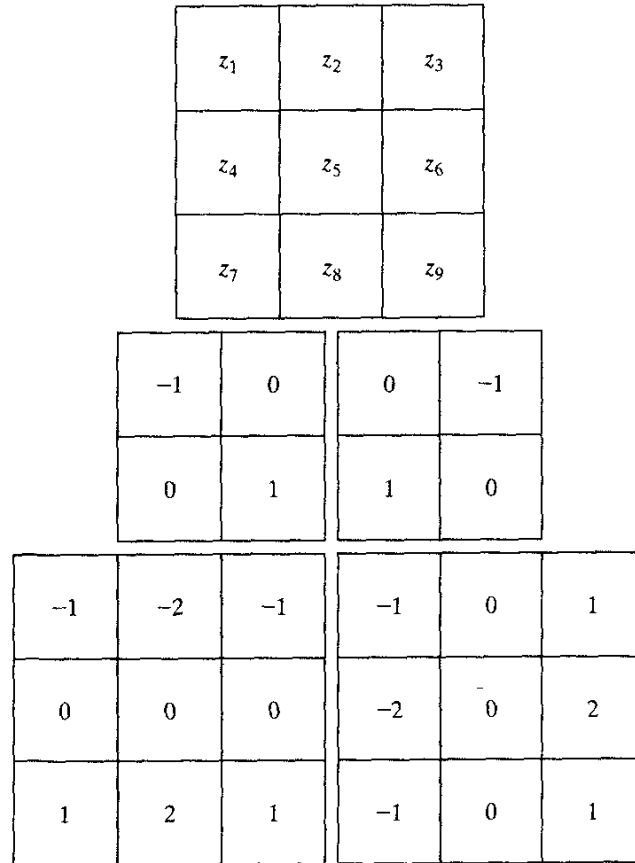
Masks of even size are awkward to implement. The smallest filter mask in which we are interested is of size  $3 \times 3$ . An approximation using absolute values, still at point  $z_5$ , but using a  $3 \times 3$  mask, is

$$\begin{aligned} \nabla f \approx & |(z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)| \\ & + |(z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)|. \end{aligned} \quad (3.7-18)$$

a  
b c  
d e

**FIGURE 3.44**

A  $3 \times 3$  region of an image (the  $z$ 's are gray-level values) and masks used to compute the gradient at point labeled  $z_5$ . All masks coefficients sum to zero, as expected of a derivative operator.

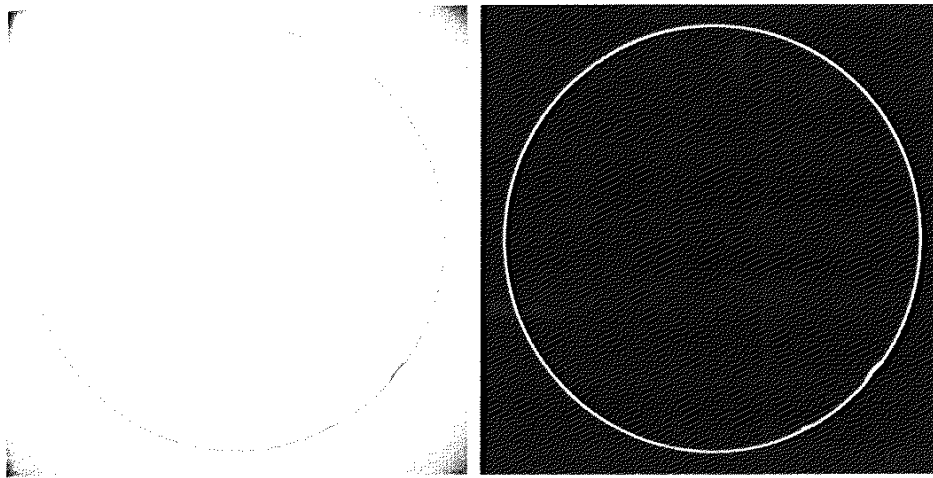


The difference between the third and first rows of the  $3 \times 3$  image region approximates the derivative in the  $x$ -direction, and the difference between the third and first columns approximates the derivative in the  $y$ -direction. The masks shown in Figs. 3.44(d) and (e), called the *Sobel operators*, can be used to implement Eq. (3.7-18) via the mechanics given in Eq. (3.5-1). The idea behind using a weight value of 2 is to achieve some smoothing by giving more importance to the center point (we discuss this in more detail in Chapter 10). Note that the coefficients in all the masks shown in Fig. 3.44 sum to 0, indicating that they would give a response of 0 in an area of constant gray level, as expected of a derivative operator.

**EXAMPLE 3.14:**  
Use of the  
gradient for edge  
enhancement.

■ The gradient is used frequently in industrial inspection, either to aid humans in the detection of defects or, what is more common, as a preprocessing step in automated inspection. We will have more to say about this in Chapters 10 and 11. However, it will be instructive at this point to consider a simple example to show how the gradient can be used to enhance defects and eliminate slowly changing background features. In this particular example, the enhancement is used as a preprocessing step for automated inspection, rather than for human analysis.

Figure 3.45(a) shows an optical image of a contact lens, illuminated by a lighting arrangement designed to highlight imperfections, such as the two edge



a b

**FIGURE 3.45**

Optical image of contact lens (note defects on the boundary at 4 and 5 o'clock).

(b) Sobel gradient. (Original image courtesy of Mr. Pete Sites, Perceptics Corporation.)

defects in the lens boundary seen at 4 and 5 o'clock. Figure 3.45(b) shows the gradient obtained using Eq. (3.7-14) with the two Sobel masks in Figs. 3.44(d) and (e). The edge defects also are quite visible in this image, but with the added advantage that constant or slowly varying shades of gray have been eliminated, thus simplifying considerably the computational task required for automated inspection. Note also that the gradient process highlighted small specs that are not readily visible in the gray-scale image (specs like these can be foreign matter, air pockets in a supporting solution, or miniscule imperfections in the lens). The ability to enhance small discontinuities in an otherwise flat gray field is another important feature of the gradient.



### Combining Spatial Enhancement Methods

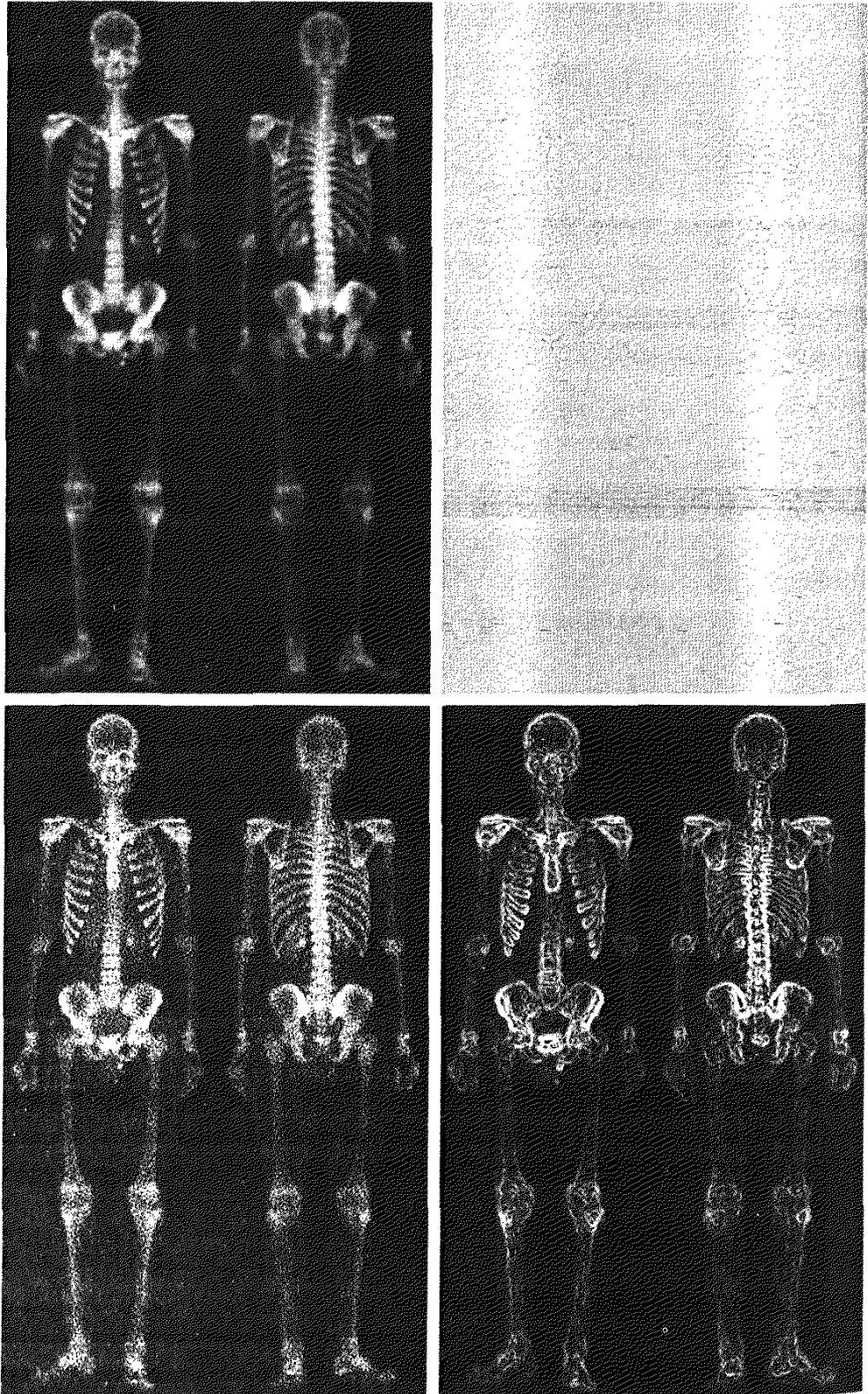
With a few exceptions, like combining blurring with thresholding in Section 3.6.1, we have focused attention thus far on individual enhancement approaches. Frequently, a given enhancement task will require application of several complementary enhancement techniques in order to achieve an acceptable result. In this section we illustrate by means of an example how to combine several of the approaches developed in this chapter to address a difficult enhancement task.

The image shown in Fig. 3.46(a) is a nuclear whole body bone scan, used to detect diseases such as bone infection and tumors. Our objective is to enhance this image by sharpening it and by bringing out more of the skeletal detail. The narrow dynamic range of the gray levels and high noise content make this image difficult to enhance. The strategy we will follow is to utilize the Laplacian to highlight fine detail, and the gradient to enhance prominent edges. For reasons that will be explained shortly, a smoothed version of the gradient image will be used to mask the Laplacian image (see Section 3.4 regarding masking). Finally, we will attempt to increase the dynamic range of the gray levels by using a gray-level transformation.

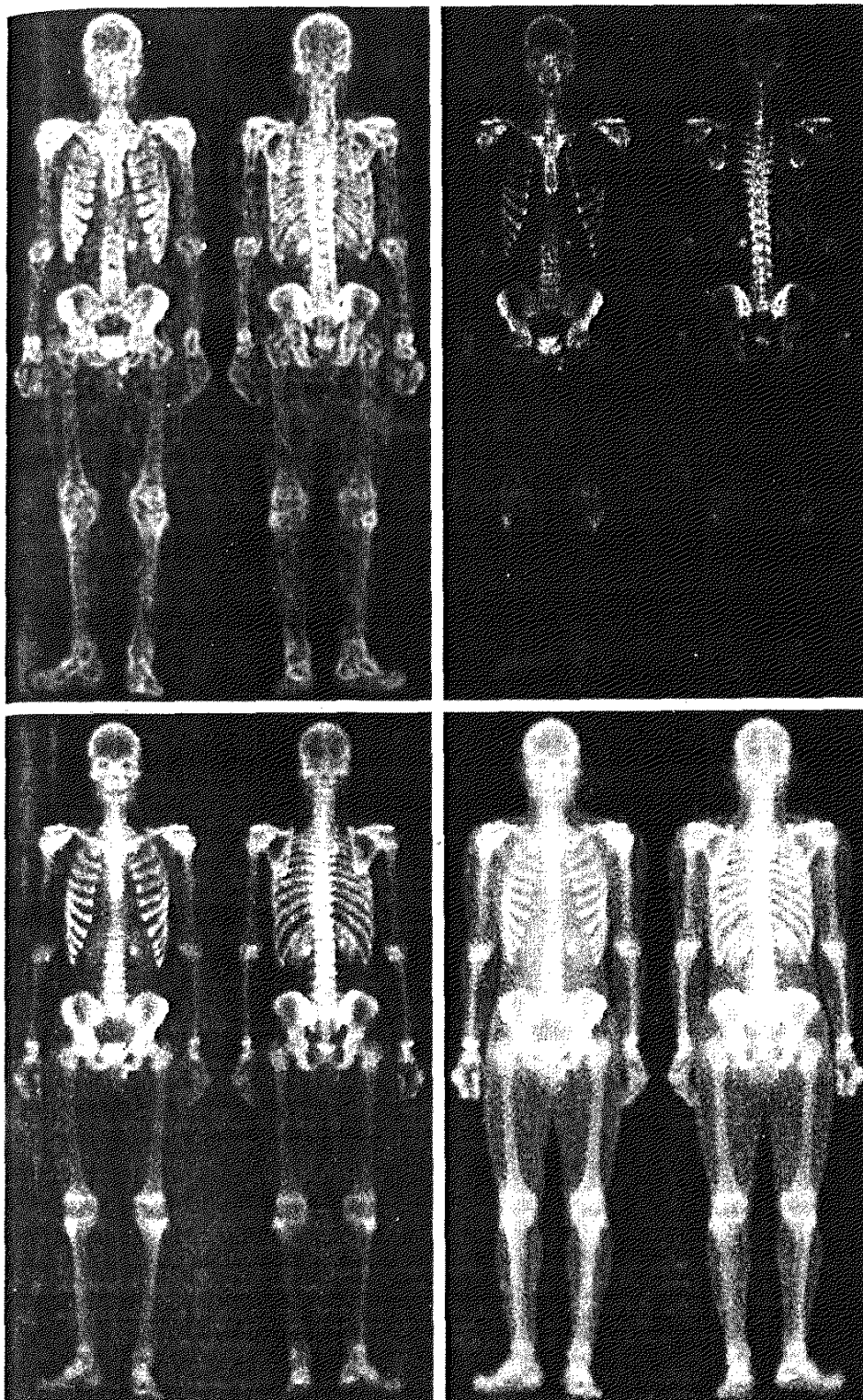
Figure 3.46 (b) shows the Laplacian of the original image, obtained using the mask in Fig. 3.39(d). This image was scaled (for display only) using the same technique as in Fig. 3.40. We can obtain a sharpened image at this point

a b  
c d

**FIGURE 3.46**  
(a) Image of whole body bone scan.  
(b) Laplacian of (a). (c) Sharpened image obtained by adding (a) and (b). (d) Sobel of (a).







e f  
g h

**FIGURE 3.46**

(Continued)

(e) Sobel image smoothed with a  $5 \times 5$  averaging filter. (f) Mask image formed by the product of (c) and (e).

(g) Sharpened image obtained by the sum of (a) and (f). (h) Final result obtained by applying a power-law transformation to (g). Compare (g) and (h) with (a). (Original image courtesy of G.E. Medical Systems.)



simply by adding Figs. 3.46(a) and (b), which are an implementation of the second line in Eq. (3.7-5) (we used a mask with a positive center coefficient). Just by looking at the noise level in (b), we would expect a rather noisy sharpened image if we added Figs. 3.46(a) and (b), a fact that is confirmed by the result shown in Fig. 3.46(c). One way that comes immediately to mind to reduce the noise is to use a median filter. However, median filtering is a non-linear process capable of removing image features. This is unacceptable in medical image processing.

An alternate approach is to use a mask formed from a smoothed version of the gradient of the original image. The motivation behind this is straightforward and is based on the properties of first- and second-order derivatives explained in Section 3.7.1. The Laplacian, being a second-order derivative operator, has the definite advantage that it is superior in enhancing fine detail. However, this causes it to produce noisier results than the gradient. This noise is most objectionable in smooth areas, where it tends to be more visible. The gradient has a stronger response in areas of significant gray-level transitions (gray-level ramps and steps) than does the Laplacian. The response of the gradient to noise and fine detail is lower than the Laplacian's and can be lowered further by smoothing the gradient with an averaging filter. The idea, then, is to smooth the gradient and multiply it by the Laplacian image. In this context, we may view the smoothed gradient as a mask image. The product will preserve details in the strong areas while reducing noise in the relatively flat areas. This process can be viewed roughly as combining the best features of the Laplacian and the gradient. The result is added to the original to obtain a final sharpened image, and could even be used in boost filtering.

Figure 3.46(d) shows the Sobel gradient of the original image, computed using Eq. (3.7-14). Components  $G_x$  and  $G_y$  were obtained using the masks in Figs. 3.44(d) and (e), respectively. As expected from our discussion in Section 3.7.1, edges are much more dominant in this image than in the Laplacian image. The smoothed gradient image shown in Fig. 3.46(e) was obtained by using an averaging filter of size  $5 \times 5$ . The two gradient images were scaled for display in the same manner as the two Laplacian images. Because the smallest possible value of a gradient image is 0, the background is black in the scaled gradient images, rather than gray as in the scaled Laplacian. The fact that Figs. 3.46(d) and (e) are much brighter than Fig. 3.46(b) is again evidence that the gradient of an image with significant edge content has values that are higher in general than in a Laplacian image.

The product of the Laplacian and smoothed-gradient image is shown in Fig. 3.46(f). Note the dominance of the strong edges and the relative lack of visible noise, which is the key objective behind masking the Laplacian with a smoothed gradient image. Adding the product image to the original resulted in the sharpened image shown in Fig. 3.46(g). The significant increase in sharpness of detail in this image over the original is evident in most parts of the image, including the ribs, spinal chord, pelvis, and skull. This type of improvement would not have been possible by using the Laplacian or gradient alone.

The sharpening procedure just discussed does not affect in an appreciable way the dynamic range of the gray levels in an image. Thus, the final step in our

enhancement task is to increase the dynamic range of the sharpened image. As we discussed in some detail in Sections 3.2 and 3.3, there are a number of gray-level transformation functions that can accomplish this objective. We do know from the results in Section 3.3.2 that histogram equalization is not likely to work well on images that have dark gray-level distributions like our images have here. Histogram specification could be a solution, but the dark characteristics of the images with which we are dealing lend themselves much better to a power-law transformation. Since we wish to spread the gray levels, the value of  $\gamma$  in Eq. (3.2-3) has to be less than 1. After a few trials with this equation we arrived at the result shown in Fig. 3.46(h), obtained with  $\gamma = 0.5$  and  $c = 1$ . Comparing this image with Fig. 3.46(g), we see that significant new detail is visible in Fig. 3.46(h). The areas around the wrists, hands, ankles, and feet are good examples of this. The skeletal bone structure also is much more pronounced, including the arm and leg bones. Note also the faint definition of the outline of the body, and of body tissue. Bringing out detail of this nature by expanding the dynamic range of the gray levels also enhanced noise, but Fig. 3.46(h) represents a significant visual improvement over the original image.

The approach just discussed is representative of the types of processes that can be linked in order to achieve results that are not possible with a single technique. The way in which the results are used depends on the application. The final user of the type of images shown in this section is likely to be a radiologist. For a number of reasons that are beyond the scope of our discussion, physicians are unlikely to rely on enhanced results to arrive at a diagnosis. However, enhanced images are quite useful in highlighting details that can serve as clues for further analysis in the original image or sequence of images. In other areas, the enhanced result may indeed be the final product. Examples are found in the printing industry, in image-based product inspection, in forensics, in microscopy, in surveillance, and in a host of other areas where the principal objective of enhancement is to obtain an image with a higher content of visual detail.

## Summary

The material presented in this chapter is representative of spatial domain techniques commonly used in practice for image enhancement. This area of image processing is a dynamic field, and new techniques and applications are reported routinely in professional literature and in new product announcements. For this reason, the topics included in this chapter were selected for their value as fundamental material that would serve as a foundation for understanding the state of the art in enhancement techniques, as well as for further study in this field. In addition to enhancement, this chapter served the purpose of introducing a number of concepts, such as filtering with spatial masks, that will be used in numerous occasions throughout the remainder of the book. In the following chapter, we deal with enhancement from a complementary viewpoint in the frequency domain. Between these two chapters, the reader will have developed a solid foundation for the terminology and some of the most fundamental tools used in image processing. The fact that these tools were introduced in the context of image enhancement is likely to aid in the understanding of how they operate on digital images.

## References and Further Reading

The material in Section 3.1 is from Gonzalez [1986]. Additional reading for the material in Section 3.2 may be found in Schowengerdt [1983], Poyton [1996], and Russ [1999]. See also the paper by Tsujii et al. [1998] regarding the optimization of image displays. Early references on histogram processing are Hummel [1974], Gonzalez and Fittes [1977], and Woods and Gonzalez [1981]. Stark [2000] gives some interesting generalizations of histogram equalization for adaptive contrast enhancement. Other approaches for contrast enhancement are exemplified by Centeno and Haertel [1997] and Cheng and Xu [2000]. For enhancement based on an ideal image model, see Highnam and Brady [1997]. For extensions of the local histogram equalization method, see Caselles et al. [1999], and Zhu et al. [1999]. See Narendra and Fitch [1981] on the use and implementation of local statistics for image enhancement. Kim et al. [1997] present an interesting approach combining the gradient with local statistics for image enhancement.

Image subtraction (Section 3.4.1) is a generic image processing tool widely used for change detection. As noted in that section, one of the principal applications of digital image subtraction is in mask mode radiography, where patient motion is a problem because motion smears the results. The problem of motion during image subtraction has received significant attention over the years, as exemplified in the survey article by Meijering et al. [1999]. The method of noise reduction by image averaging (Section 3.4.2) was first proposed by Kohler and Howell [1963]. See Peebles [1993] regarding the expected value of the mean and variance of a sum of random variables.

For additional reading on linear spatial filters and their implementation, see Umbaugh [1998], Jain [1989], and Rosenfeld and Kak [1982]. Rank-order filters are discussed in these references as well. Wilburn [1998] discusses generalizations of rank-order filters. The book by Pitas and Venetsanopoulos [1990] also deals with median and other nonlinear spatial filters. A special issue of *IEEE Transactions in Image Processing* [1996] is dedicated to the topic of nonlinear image processing. The material on high-boost filtering is from Schowengerdt [1983]. We will encounter again many of the spatial filters introduced in this chapter in discussions dealing with image restoration (Chapter 5) and edge detection (Chapter 10).

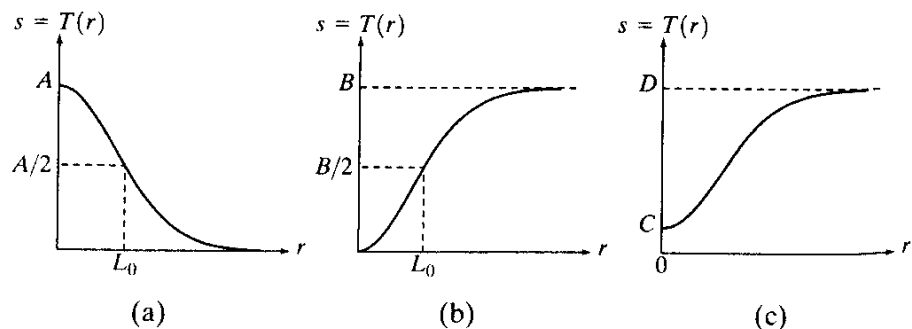
## Problems

- 3.1** Exponentials of the form  $e^{-\alpha r^2}$ , with  $\alpha$  a positive constant, are useful for constructing smooth gray-level transformation functions. Start with this basic function and construct transformation functions having the general shapes shown in the following figures. The constants shown are *input* parameters, and your proposed transformations must include them in their specification. (For simplicity in your answers,  $L_0$  is not a required parameter in the third curve.)



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.



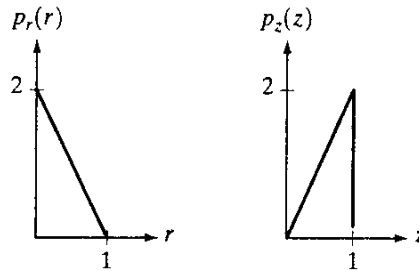
- 3.2 ★ (a)** Give a continuous function for implementing the contrast stretching transformation shown in Fig. 3.2(a). In addition to  $m$ , your function must include a parameter,  $E$ , for controlling the slope of the function as it transitions from low to high gray-level values. Your function should be normalized so that its minimum and maximum values are 0 and 1, respectively.
- (b)** Sketch a family of transformations as a function of parameter  $E$ , for a fixed value  $m = L/2$ , where  $L$  is the number of gray levels in the image.
- (c)** What is the smallest value of  $E$  that will make your function *effectively* perform as the function in Fig. 3.2(b)? In other words, your function does not have to be identical to Fig. 3.2(b). It just has to yield the same result of producing a binary image. Assume that you are working with 8-bit images, and let  $m = 128$ . Also, let  $C$  be the smallest positive number representable in the computer you are using.
- 3.3** Propose a set of gray-level-slicing transformations capable of producing all the individual bit planes of an 8-bit monochrome image. (For example, a transformation function with the property  $T(r) = 0$  for  $r$  in the range  $[0, 127]$ , and  $T(r) \approx 255$  for  $r$  in the range  $[128, 255]$  produces an image of the 7th bit plane in an 8-bit image.)
- 3.4 ★ (a)** What effect would setting to zero the lower-order bit planes have on the histogram of an image in general?
- (b)** What would be the effect on the histogram if we set to zero the higher-order bit planes instead?
- ★ 3.5** Explain why the discrete histogram equalization technique does not, in general, yield a flat histogram.
- 3.6** Suppose that a digital image is subjected to histogram equalization. Show that a second pass of histogram equalization will produce exactly the same result as the first pass.
- 3.7** In some applications it is useful to model the histogram of input images as Gaussian probability density functions of the form

$$p_r(r) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(r-m)^2}{2\sigma^2}}$$

where  $m$  and  $\sigma$  are the mean and standard deviation of the Gaussian PDF. The approach is to let  $m$  and  $\sigma$  be measures of average gray level and contrast of a given image. What is the transformation function you would use for histogram equalization?

- ★ 3.8** Assuming continuous values, show by example that it is possible to have a case in which the transformation function given in Eq. (3.3-4) satisfies Conditions (a) and (b) in Section 3.3.1, but its inverse may fail to be single valued.
- 3.9 (a)** Show that the discrete transformation function given in Eq. (3.3-8) for histogram equalization satisfies conditions (a) and (b) in Section 3.3.1.
- (b)** Show by example that this does not hold in general for the inverse discrete transformation function given in Eq. (3.3-9).
- ★ (c)** Show that the inverse discrete transformation in Eq. (3.3-9) satisfies Conditions (a) and (b) in Section 3.3.1 if none of the gray levels  $r_k$ ,  $k = 0, 1, \dots, L - 1$ , are missing.

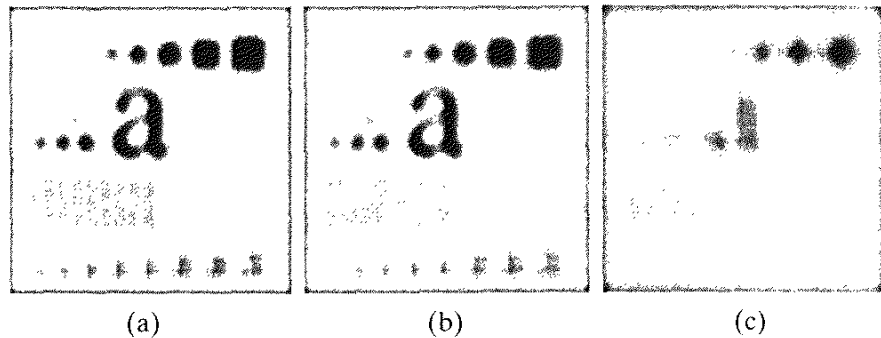
- 3.10** An image has the gray level PDF  $p_r(r)$  shown in the following diagram. It is desired to transform the gray levels of this image so that they will have the specified  $p_z(z)$  shown. Assume continuous quantities and find the transformation (in terms of  $r$  and  $z$ ) that will accomplish this.



- ★ **3.11** Propose a method for updating the local histogram for use in the local enhancement technique discussed in Section 3.3.3.
- 3.12** Two images,  $f(x, y)$  and  $g(x, y)$ , have histograms  $h_f$  and  $h_g$ . Give the conditions under which you can determine the histograms of
- (a)  $f(x, y) + g(x, y)$
  - (b)  $f(x, y) - g(x, y)$
  - (c)  $f(x, y) \times g(x, y)$
  - (d)  $f(x, y) \div g(x, y)$
- in terms of  $h_f$  and  $h_g$ . Explain how to obtain the histogram in each case.
- 3.13** Consider two 8-bit images whose gray levels span the full range from 0 to 255.
- (a) Discuss the limiting effect of repeatedly subtracting image (b) from image (a).
  - (b) Would reversing the order of the images yield a different result?
- ★ **3.14** Image subtraction is used often in industrial applications for detecting missing components in product assembly. The approach is to store a “golden” image that corresponds to a correct assembly; this image is then subtracted from incoming images of the same product. Ideally, the differences would be zero if the new products are assembled correctly. Difference images for products with missing components would be nonzero in the area where they differ from the golden image. What conditions do you think have to be met in practice for this method to work?
- 3.15** Prove the validity of Eqs. (3.4-4) and (3.4-5).
- 3.16** In an industrial application, X-ray imaging is to be used to inspect the inside of certain composite castings. The objective is to look for voids in the castings, which typically appear as small blobs in the image. However, due to properties in of the casting material and X-ray energy used, high noise content often makes inspection difficult, so the decision is made to use image averaging to reduce the noise and thus improve visible contrast. In computing the average, it is important to keep the number of images as small as possible to reduce the time the parts have to remain stationary during imaging. After numerous experiments, it is concluded that decreasing the noise variance by a factor of 10 is sufficient. If the imaging device can produce 30 frames/s, how long would the castings have to remain stationary during imaging to achieve the desired decrease in variance? Assume that the noise is uncorrelated and has zero mean.

- 3.17** The implementation of linear spatial filters requires moving the center of a mask throughout an image and, at each location, computing the sum of products of the mask coefficients with the corresponding pixels at that location (see Section 3.5). In the case of lowpass filtering, all coefficients are 1, allowing use of a so-called *box-filter* or *moving-average* algorithm, which consists of updating only the part of the computation that changes from one location to the next.
- ★ **(a)** Formulate such an algorithm for an  $n \times n$  filter, showing the nature of the computations involved and the scanning sequence used for moving the mask around the image.
  - (b)** The ratio of the number of computations performed by a brute-force implementation to the number of computations performed by the box-filter algorithm is called the *computational advantage*. Obtain the computational advantage in this case and plot it as a function of  $n$  for  $n > 1$ . The  $1/n^2$  scaling factor is common to both approaches, so you need not consider it in obtaining the computational advantage. Assume that the image has an outer border of zeros that is thick enough to allow you to ignore border effects in your analysis.
- 3.18** Discuss the limiting effect of repeatedly applying a  $3 \times 3$  lowpass spatial filter to a digital image. You may ignore border effects.
- 3.19** ★ **(a)** It was stated in Section 3.6.2 that isolated clusters of dark or light (with respect to the background) pixels whose area is less than one-half the area of a median filter are eliminated (forced to the median value of the neighbors) by the filter. Assume a filter of size  $n \times n$ , with  $n$  odd, and explain why this is so.
- (b)** Consider an image having various sets of pixel clusters. Assume that all points in a cluster are lighter or darker than the background (but not both simultaneously in the same cluster), and that the area of each cluster is less than or equal to  $n^2/2$ . In terms of  $n$ , under what condition would one or more of these clusters cease to be isolated in the sense described in part (a)?
- ★ **3.20** **(a)** Develop a procedure for computing the median of an  $n \times n$  neighborhood.
- (b)** Propose a technique for updating the median as the center of the neighborhood is moved from pixel to pixel.
- 3.21** **(a)** In a character recognition application, text pages are reduced to binary form using a thresholding transformation function of the form shown in Fig. 3.2(b). This is followed by a procedure that thins the characters until they become strings of binary 1's on a background of 0's. Due to noise, the binarization and thinning processes result in broken strings of characters with gaps ranging from 1 to 3 pixels. One way to "repair" the gaps is to run an averaging mask over the binary image to blur it, and thus create bridges of nonzero pixels between gaps. Give the (odd) size of the smallest averaging mask capable of performing this task.
- (b)** After bridging the gaps, it is desired to threshold the image in order to convert it back to binary form. For your answer in (a), what is the minimum value of the threshold required to accomplish this, without causing the segments to break up again?
- ★ **3.22** The three images shown were blurred using square averaging masks of sizes  $n = 23, 25$ , and  $45$ , respectively. The vertical bars on the left lower part of (a) and (c) are blurred, but a clear separation exists between them. However, the bars

have merged in image (b), in spite of the fact that the mask that produced this image is significantly smaller than the mask that produced image (c). Explain this.



- 3.23** Consider an application such as the one shown in Fig. 3.36, in which it is desired to eliminate objects smaller than those enclosed in a square of size  $q \times q$  pixels. Suppose that we want to reduce the average gray level of those objects to one-tenth of their original average gray level. In this way, those objects will be closer to the gray level of the background and they can then be eliminated by thresholding. Give the (odd) size of the smallest averaging mask that will accomplish the desired reduction in average gray level in only one pass of the mask over the image.
- 3.24** In a given application an averaging mask is applied to input images to reduce noise, and then a Laplacian mask is applied to enhance small details. Would the result be the same if the order of these operations were reversed?
- ★ **3.25** Show that the Laplacian operation defined in Eq. (3.7-1) is isotropic (invariant to rotation). You will need the following equations relating coordinates after axis rotation by an angle  $\theta$ :

$$x = x' \cos \theta - y' \sin \theta$$

$$y = x' \sin \theta + y' \cos \theta$$

where  $(x, y)$  are the unrotated and  $(x', y')$  are the rotated coordinates.

- 3.26** Give a  $3 \times 3$  mask for performing unsharp masking in a single pass through an image.
- ★ **3.27** Show that subtracting the Laplacian from an image is proportional to unsharp masking. Use the definition for the Laplacian given in Eq. (3.7-4).
- 3.28** (a) Show that the magnitude of the gradient given in Eq. (3.7-13) is an isotropic operation. (See Problem 3.25.)  
 (b) Show that the isotropic property is lost in general if the gradient is computed using Eq. (3.7-14).
- 3.29** A CCD TV camera is used to perform a long-term study by observing the same area 24 hours a day, for 30 days. Digital images are captured and transmitted to a central location every 5 minutes. The illumination of the scene changes from natural daylight to artificial lighting. At no time is the scene without illumination, so it is always possible to obtain an image. Because the range of illumination is such that it is always in the linear operating range of the camera, it is decided not to employ any compensating mechanisms on the camera itself. Rather, it is decided to use digital techniques to postprocess, and thus normalize, the images to the equivalent of constant illumination. Propose a method to do this. You are at liberty to use any method you wish, but state clearly all the assumptions you made in arriving at your design.