3 *Image Enhancement in the Spatial Domain*

> It makes all the difference whether one sees darkness through the light or brightness through the shadows.
>
> *David Lindsay*

Preview

The principal objective of enhancement is to process an image so that the result is more suitable than the original image for a specific application. The word *specific* is important, because it establishes at the outset that the techniques discussed in this chapter are very much problem oriented. Thus, for example, a method that is quite useful for enhancing X-ray images may not necessarily be the best approach for enhancing pictures of Mars transmitted by a space probe. Regardless of the method used, however, image enhancement is one of the most interesting and visually appealing areas of image processing.

Image enhancement approaches fall into two broad categories: spatial domain methods and frequency domain methods. The term *spatial domain* refers to the image plane itself, and approaches in this category are based on direct manipulation of pixels in an image. *Frequency domain* processing techniques are based on modifying the Fourier transform of an image. Spatial methods are covered in this chapter, and frequency domain enhancement is discussed in Chapter 4. Enhancement techniques based on various combinations of methods from these two categories are not unusual. We note also that many of the fundamental techniques introduced in this chapter in the context of enhancement are used in subsequent chapters for a variety of other image processing applications.

There is no general theory of image enhancement. When an image is processed for visual interpretation, the viewer is the ultimate judge of how well

a particular method works. Visual evaluation of image quality is a highly subjective process, thus making the definition of a "good image" an elusive standard by which to compare algorithm performance. When the problem is one of processing images for machine perception, the evaluation task is somewhat easier. For example, in dealing with a character recognition application, and leaving aside other issues such as computational requirements, the best image processing method would be the one yielding the best machine recognition results. However, even in situations when a clear-cut criterion of performance can be imposed on the problem, a certain amount of trial and error usually is required before a particular image enhancement approach is selected.

3.1 Background

As indicated previously, the term *spatial domain* refers to the aggregate of pixels composing an image. Spatial domain methods are procedures that operate directly on these pixels. Spatial domain processes will be denoted by the expression

$$g(x, y) = T[f(x, y)] \tag{3.1-1}$$

where $f(x, y)$ is the input image, $g(x, y)$ is the processed image, and T is an operator on f, defined over some neighborhood of (x, y). In addition, T can operate on a *set* of input images, such as performing the pixel-by-pixel sum of K images for noise reduction, as discussed in Section 3.4.2.

The principal approach in defining a neighborhood about a point (x, y) is to use a square or rectangular subimage area centered at (x, y), as Fig. 3.1 shows. The center of the subimage is moved from pixel to pixel starting, say, at the top left corner. The operator T is applied at each location (x, y) to yield the output, g, at that location. The process utilizes only the pixels in the area of the image spanned by the neighborhood. Although other neighborhood shapes, such as ap-

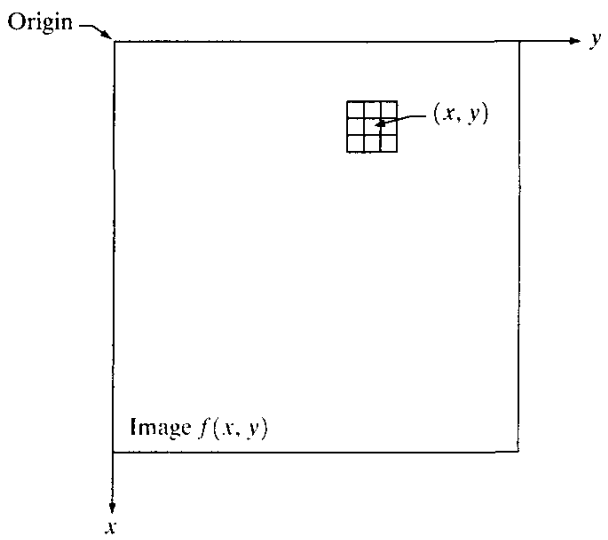


FIGURE 3.1 A 3×3 neighborhood about a point (x, y) in an image.

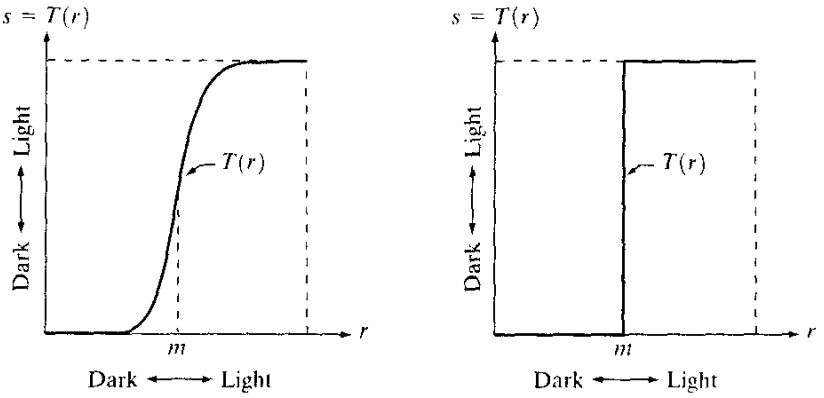
proximations to a circle, sometimes are used, square and rectangular arrays are by far the most predominant because of their ease of implementation.

The simplest form of T is when the neighborhood is of size 1×1 (that is, a single pixel). In this case, g depends only on the value of f at (x, y), and T becomes a *gray-level* (also called an *intensity* or *mapping*) *transformation function* of the form

$$s = T(r) \tag{3.1-2}$$

where, for simplicity in notation, r and s are variables denoting, respectively, the gray level of $f(x, y)$ and $g(x, y)$ at any point (x, y). For example, if $T(r)$ has the form shown in Fig. 3.2(a), the effect of this transformation would be to produce an image of higher contrast than the original by darkening the levels below m and brightening the levels above m in the original image. In this technique, known as *contrast stretching*, the values of r below m are compressed by the transformation function into a narrow range of s, toward black. The opposite effect takes place for values of r above m. In the limiting case shown in Fig. 3.2(b), $T(r)$ produces a two-level (binary) image. A mapping of this form is called a *thresholding* function. Some fairly simple, yet powerful, processing approaches can be formulated with gray-level transformations. Because enhancement at any point in an image depends only on the gray level at that point, techniques in this category often are referred to as *point processing*.

Larger neighborhoods allow considerably more flexibility. The general approach is to use a function of the values of f in a predefined neighborhood of (x, y) to determine the value of g at (x, y). One of the principal approaches in this formulation is based on the use of so-called *masks* (also referred to as *filters, kernels, templates*, or *windows*). Basically, a mask is a small (say, 3×3) 2-D array, such as the one shown in Fig. 3.1, in which the values of the mask coefficients determine the nature of the process, such as image sharpening. Enhancement techniques based on this type of approach often are referred to as *mask processing* or *filtering*. These concepts are discussed in Section 3.5.



a b

FIGURE 3.2 Gray-level transformation functions for contrast enhancement.

Some Basic Gray Level Transformations

We begin the study of image enhancement techniques by discussing gray-level transformation functions. These are among the simplest of all image enhancement techniques. The values of pixels, before and after processing, will be denoted by r and s, respectively. As indicated in the previous section, these values are related by an expression of the form $s = T(r)$, where T is a transformation that maps a pixel value r into a pixel value s. Since we are dealing with digital quantities, values of the transformation function typically are stored in a one-dimensional array and the mappings from r to s are implemented via table lookups. For an 8-bit environment, a lookup table containing the values of T will have 256 entries.

As an introduction to gray-level transformations, consider Fig. 3.3, which shows three basic types of functions used frequently for image enhancement: linear (negative and identity transformations), logarithmic (log and inverse-log transformations), and power-law (nth power and nth root transformations). The identity function is the trivial case in which output intensities are identical to input intensities. It is included in the graph only for completeness.

3.2.1 Image Negatives

The negative of an image with gray levels in the range $[0, L-1]$ is obtained by using the negative transformation shown in Fig. 3.3, which is given by the expression

$$s = L - 1 - r. \tag{3.2-1}$$

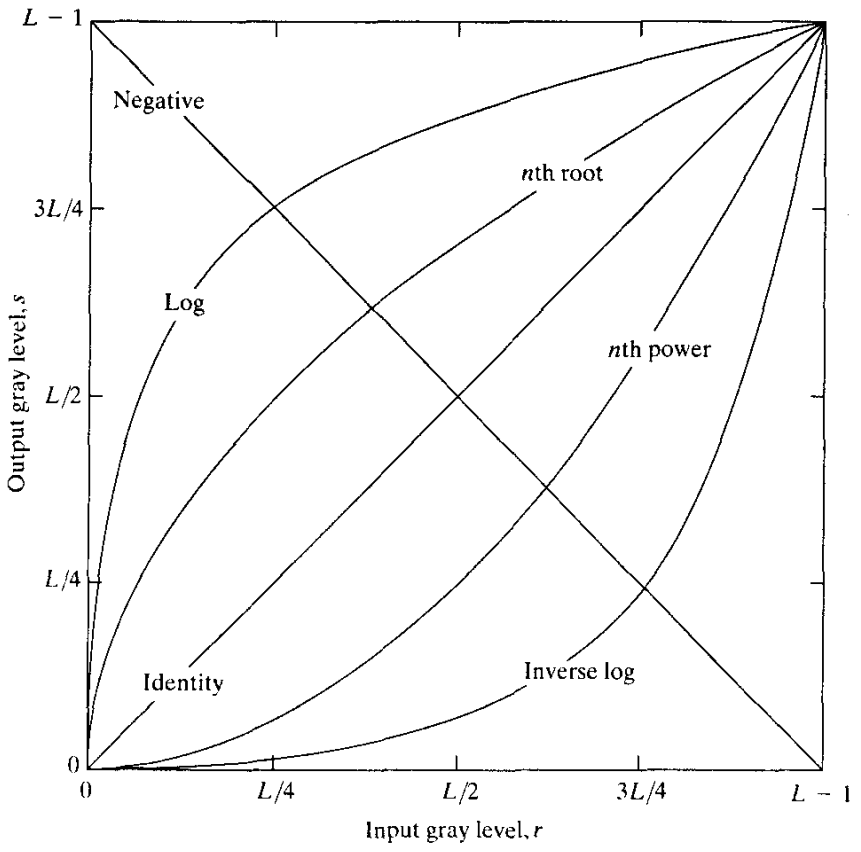
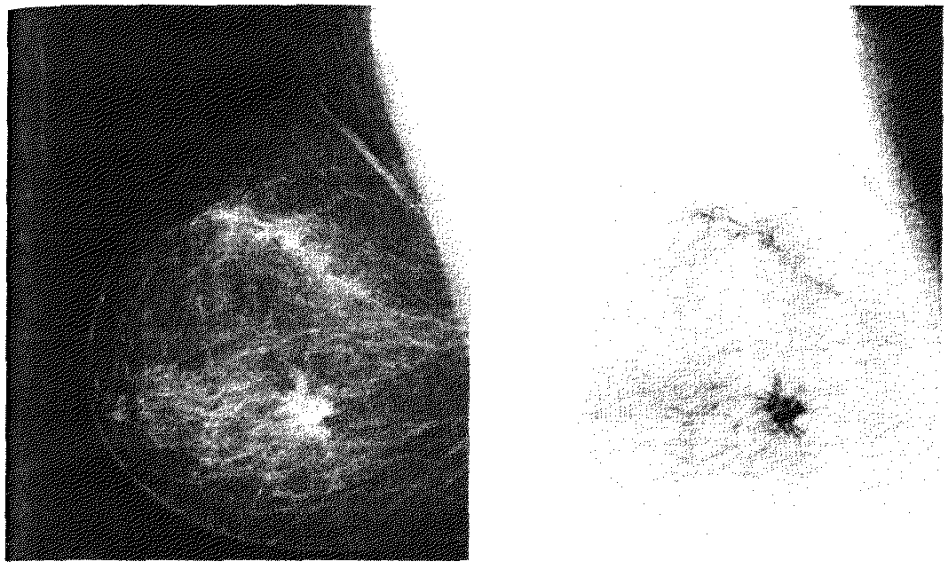


FIGURE 3.3 Some basic gray-level transformation functions used for image enhancement.



a b

FIGURE 3.4 (a) Original digital mammogram. (b) Negative image obtained using the negative transformation in Eq. (3.2-1). (Courtesy of G.E. Medical Systems.)

Reversing the intensity levels of an image in this manner produces the equivalent of a photographic negative. This type of processing is particularly suited for enhancing white or gray detail embedded in dark regions of an image, especially when the black areas are dominant in size. An example is shown in Fig. 3.4. The original image is a digital mammogram showing a small lesion. In spite of the fact that the visual content is the same in both images, note how much easier it is to analyze the breast tissue in the negative image in this particular case.

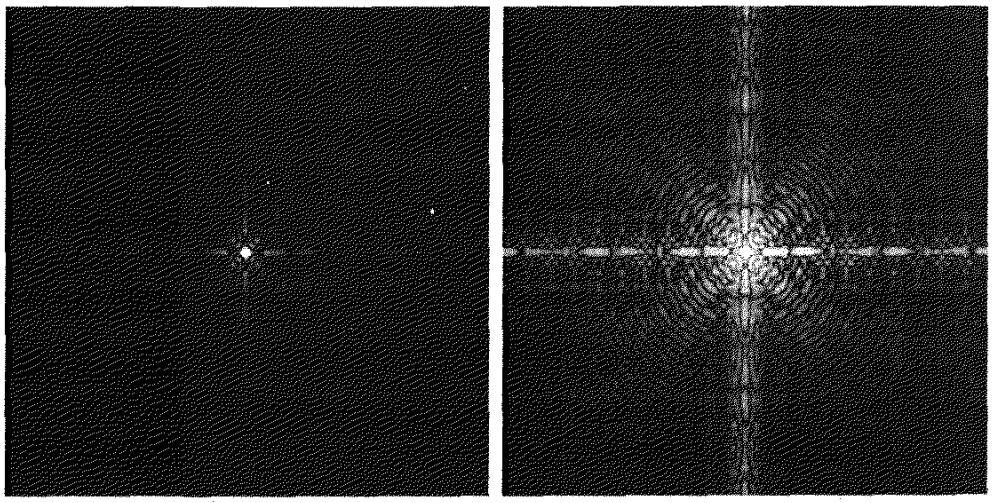
3.2.2 Log Transformations

The general form of the log transformation shown in Fig. 3.3 is

$$s = c \log(1 + r) \tag{3.2-2}$$

where c is a constant, and it is assumed that $r \geq 0$. The shape of the log curve in Fig. 3.3 shows that this transformation maps a narrow range of low gray-level values in the input image into a wider range of output levels. The opposite is true of higher values of input levels. We would use a transformation of this type to expand the values of dark pixels in an image while compressing the higher-level values. The opposite is true of the inverse log transformation.

Any curve having the general shape of the log functions shown in Fig. 3.3 would accomplish this spreading/compressing of gray levels in an image. In fact, the power-law transformations discussed in the next section are much more versatile for this purpose than the log transformation. However, the log function has the important characteristic that it compresses the dynamic range of images with large variations in pixel values. A classic illustration of an application in which pixel values have a large dynamic range is the Fourier spectrum, which will be discussed in Chapter 4. At the moment, we are concerned only with the image characteristics of spectra. It is not unusual to encounter spectrum values



a b

FIGURE 3.5
(a) Fourier spectrum.
(b) Result of applying the log transformation given in Eq. (3.2-2) with $c = 1$.

that range from 0 to 10^6 or higher. While processing numbers such as these presents no problems for a computer, image display systems generally will not be able to reproduce faithfully such a wide range of intensity values. The net effect is that a significant degree of detail will be lost in the display of a typical Fourier spectrum.

As an illustration of log transformations, Fig. 3.5(a) shows a Fourier spectrum with values in the range 0 to 1.5×10^6. When these values are scaled linearly for display in an 8-bit system, the brightest pixels will dominate the display, at the expense of lower (and just as important) values of the spectrum. The effect of this dominance is illustrated vividly by the relatively small area of the image in Fig. 3.5(a) that is not perceived as black. If, instead of displaying the values in this manner, we first apply Eq. (3.2-2) (with $c = 1$ in this case) to the spectrum values, then the range of values of the result become 0 to 6.2, a more manageable number. Figure 3.5(b) shows the result of scaling this new range linearly and displaying the spectrum in the same 8-bit display. The wealth of detail visible in this image as compared to a straight display of the spectrum is evident from these pictures. Most of the Fourier spectra seen in image processing publications have been scaled in just this manner.

3.2.2 Power-Law Transformations

Power-law transformations have the basic form

$$s = cr^{\gamma} \tag{3.2-3}$$

where c and γ are positive constants. Sometimes Eq. (3.2-3) is written as $s = c(r + \varepsilon)^{\gamma}$ to account for an offset (that is, a measurable output when the input is zero). However, offsets typically are an issue of display calibration and as a result they are normally ignored in Eq. (3.2-3). Plots of s versus r for various values of γ are shown in Fig. 3.6. As in the case of the log transformation, power-law curves with fractional values of γ map a narrow range of dark input values into a wider range of output values, with the opposite being true for high-

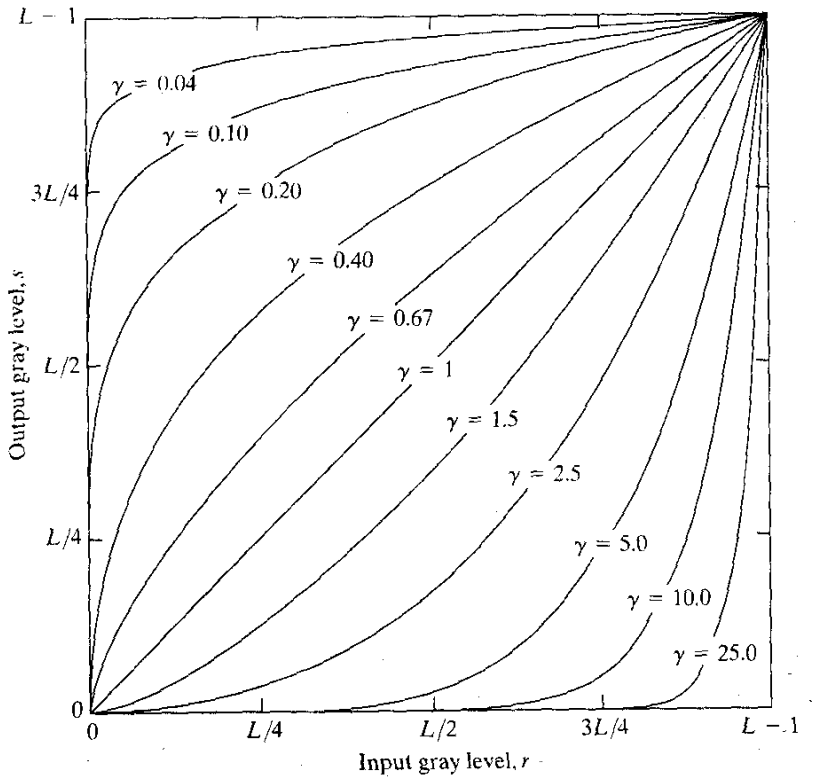
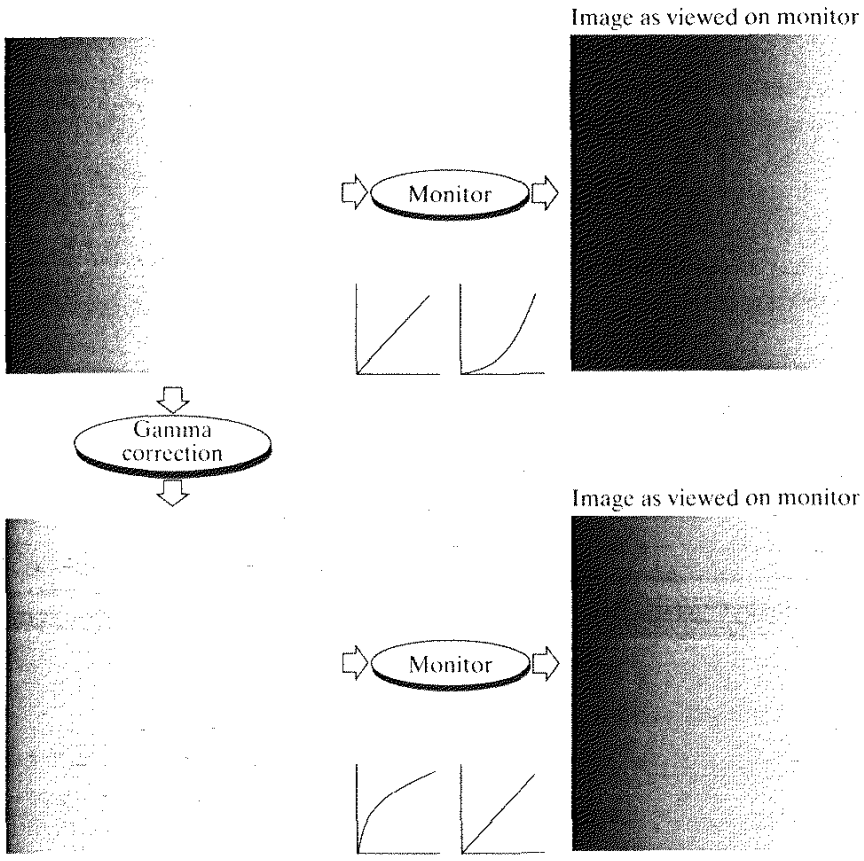


FIGURE 3.6 Plots of the equation $s = cr^{\gamma}$ for various values of γ ($c = 1$ in all cases).

er values of input levels. Unlike the log function, however, we notice here a family of possible transformation curves obtained simply by varying γ. As expected, we see in Fig. 3.6 that curves generated with values of $\gamma > 1$ have exactly the opposite effect as those generated with values of $\gamma < 1$. Finally, we note that Eq. (3.2-3) reduces to the identity transformation when $c = \gamma = 1$.

A variety of devices used for image capture, printing, and display respond according to a power law. By convention, the exponent in the power-law equation is referred to as *gamma* [hence our use of this symbol in Eq. (3.2-3)]. The process used to correct this power-law response phenomena is called *gamma correction*. For example, cathode ray tube (CRT) devices have an intensity-to-voltage response that is a power function, with exponents varying from approximately 1.8 to 2.5. With reference to the curve for $\gamma = 2.5$ in Fig. 3.6, we see that such display systems would tend to produce images that are darker than intended. This effect is illustrated in Fig. 3.7. Figure 3.7(a) shows a simple gray-scale linear wedge input into a CRT monitor. As expected, the output of the monitor appears darker than the input, as shown in Fig. 3.7(b). Gamma correction in this case is straightforward. All we need to do is preprocess the input image before inputting it into the monitor by performing the transformation $s = r^{1/2.5} = r^{0.4}$. The result is shown in Fig. 3.7(c). When input into the same monitor, this gamma-corrected input produces an output that is close in appearance to the original image, as shown in Fig. 3.7(d). A similar analysis would



a b
c d

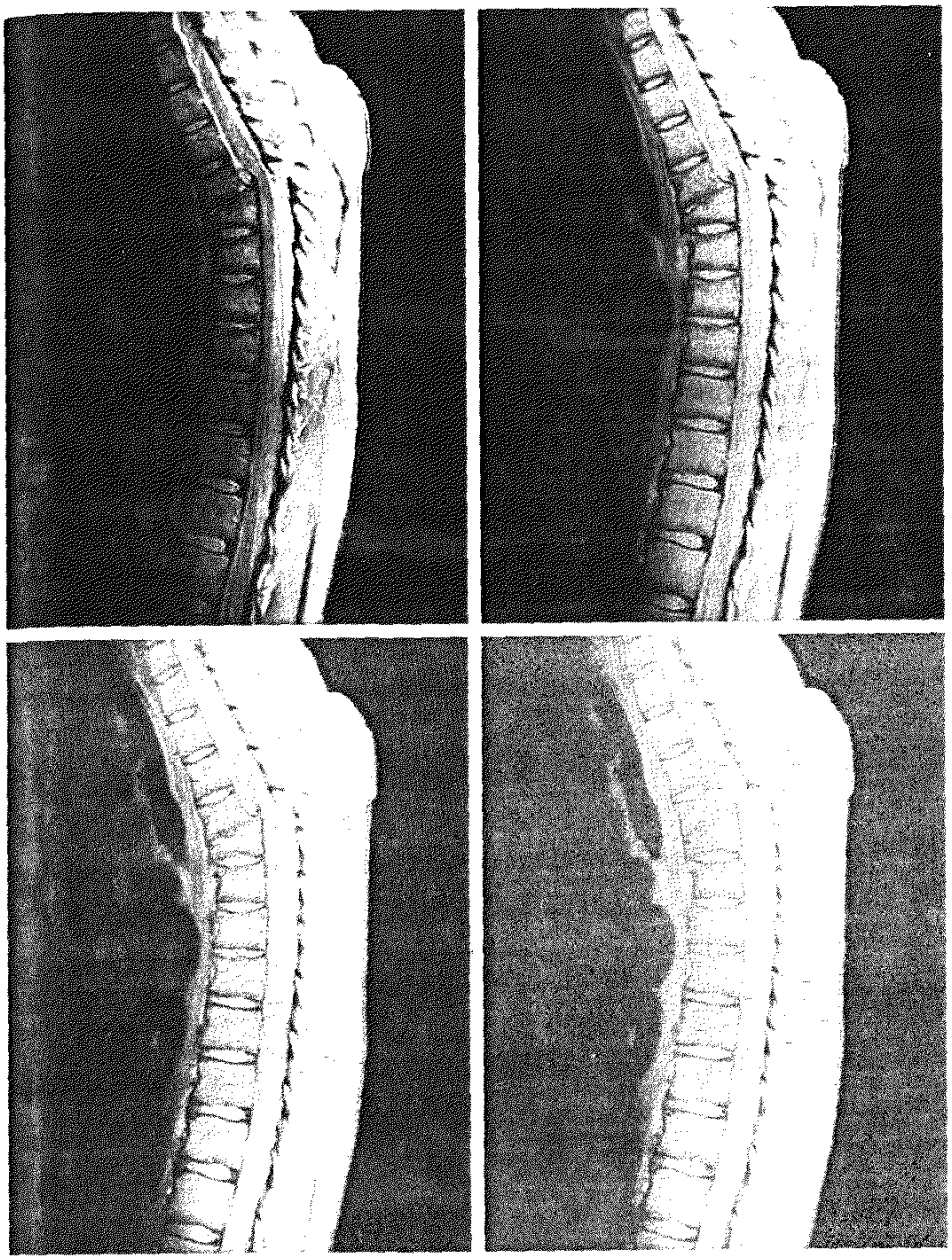
FIGURE 3.7
(a) Linear-wedge gray-scale image. (b) Response of monitor to linear wedge. (c) Gamma-corrected wedge. (d) Output of monitor.

apply to other imaging devices such as scanners and printers. The only difference would be the device-dependent value of gamma (Poynton [1996]).

Gamma correction is important if displaying an image accurately on a computer screen is of concern. Images that are not corrected properly can look either bleached out, or, what is more likely, too dark. Trying to reproduce colors accurately also requires some knowledge of gamma correction because varying the value of gamma correction changes not only the brightness, but also the ratios of red to green to blue. Gamma correction has become increasingly important in the past few years, as use of digital images for commercial purposes over the Internet has increased. It is not unusual that images created for a popular Web site will be viewed by millions of people, the majority of whom will have different monitors and/or monitor settings. Some computer systems even have partial gamma correction built in. Also, current image standards do not contain the value of gamma with which an image was created, thus complicating the issue further. Given these constraints, a reasonable approach when storing images in a Web site is to preprocess the images with a gamma that represents an "average" of the types of monitors and computer systems that one expects in the open market at any given point in time.

EXAMPLE 3.1: Contrast enhancement using power-law transformations.

In addition to gamma correction, power-law transformations are useful for general-purpose contrast manipulation. Figure 3.8(a) shows a magnetic resonance (MR) image of an upper thoracic human spine with a fracture dislocation



a b
c d

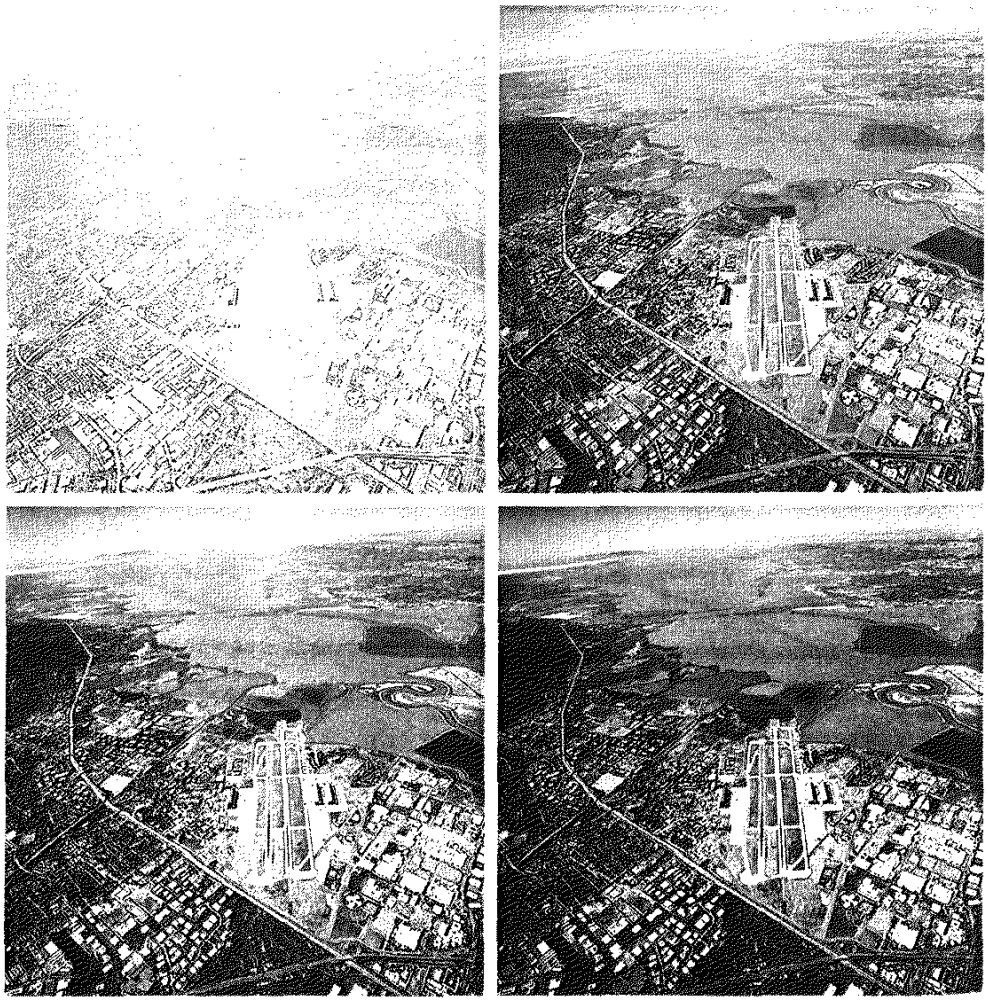
FIGURE 3.8
(a) Magnetic resonance (MR) image of a fractured human spine.
(b)–(d) Results of applying the transformation in Eq. (3.2-3) with $c = 1$ and $\gamma = 0.6, 0.4$, and 0.3, respectively. (Original image for this example courtesy of Dr. David R. Pickens, Department of Radiology and Radiological Sciences, Vanderbilt University Medical Center.)

and spinal cord impingement. The fracture is visible near the vertical center of the spine, approximately one-fourth of the way down from the top of the picture. Since the given image is predominantly dark, an expansion of gray levels are desirable. This can be accomplished with a power-law transformation with a fractional exponent. The other images shown in the figure were obtained by processing Fig. 3.8(a) with the power-law transformation function of Eq. (3.2-3). The values of gamma corresponding to images (b) through (d) are 0.6, 0.4, and 0.3, respectively (the value of c was 1 in all cases). We note that, as gamma decreased from 0.6 to 0.4, more detail became visible. A further decrease of gamma

to 0.3 enhanced a little more detail in the background, but began to reduce contrast to the point where the image started to have a very slight "washed-out" look, especially in the background. By comparing all results, we see that the best enhancement in terms of contrast and discernable detail was obtained with $\gamma = 0.4$. A value of $\gamma = 0.3$ is an approximate limit below which contrast in this particular image would be reduced to an unacceptable level.

EXAMPLE 3.2: Another illustration of power-law transformations.

Figure 3.9(a) shows the opposite problem of Fig. 3.8(a). The image to be enhanced now has a washed-out appearance, indicating that a compression of gray levels is desirable. This can be accomplished with Eq. (3.2-3) using values of γ greater than 1. The results of processing Fig. 3.9(a) with $\gamma = 3.0$, 4.0, and 5.0 are shown in Figs. 3.9(b) through (d). Suitable results were obtained with gamma values of 3.0 and 4.0, the latter having a slightly more appealing appearance because it has higher contrast. The result obtained with $\gamma = 5.0$ has areas that are too dark, in which some detail is lost. The dark region to the left of the main road in the upper left quadrant is an example of such an area.



a b
c d

FIGURE 3.9
(a) Aerial image. (b)–(d) Results of applying the transformation in Eq. (3.2-3) with $c = 1$ and $\gamma = 3.0$, 4.0, and 5.0, respectively. (Original image for this example courtesy of NASA.)

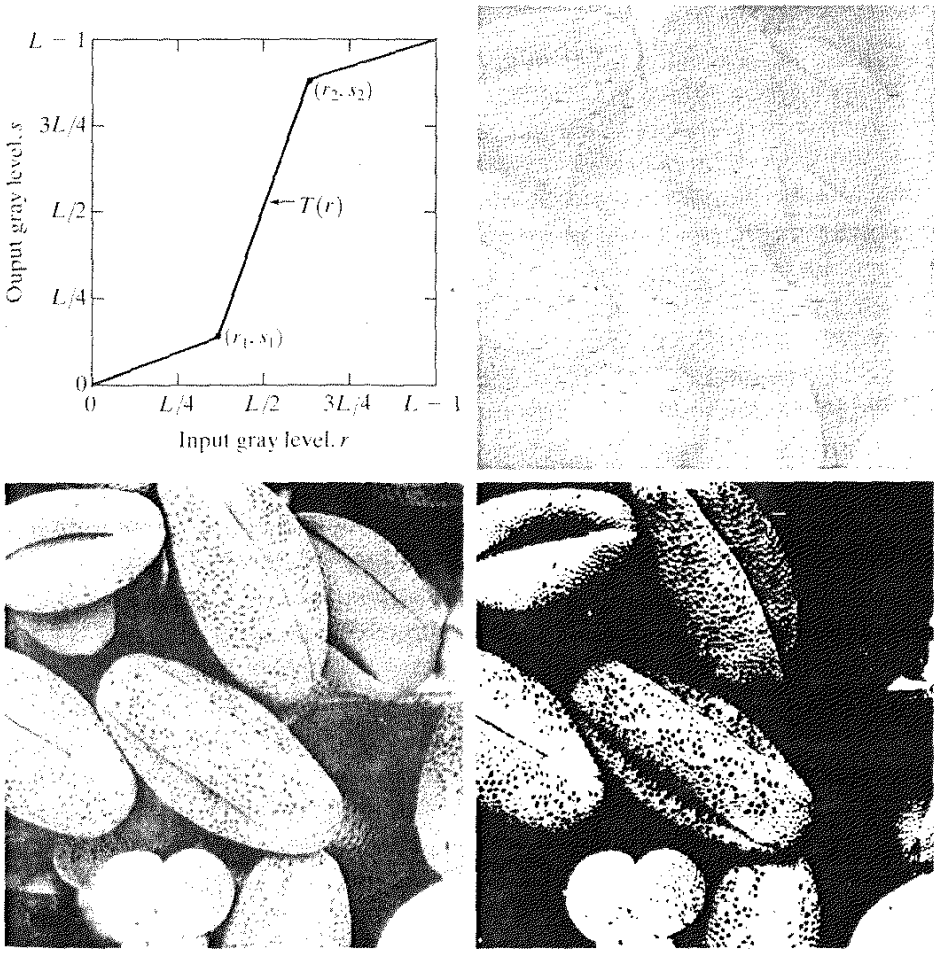
Piecewise-Linear Transformation Functions

A complementary approach to the methods discussed in the previous three sections is to use piecewise linear functions. The principal advantage of piecewise linear functions over the types of functions we have discussed thus far is that the form of piecewise functions can be arbitrarily complex. In fact, as we will see shortly, a practical implementation of some important transformations can be formulated only as piecewise functions. The principal disadvantage of piecewise functions is that their specification requires considerably more user input.

Contrast stretching

One of the simplest piecewise linear functions is a contrast-stretching transformation. Low-contrast images can result from poor illumination, lack of dynamic range in the imaging sensor, or even wrong setting of a lens aperture during image acquisition. The idea behind contrast stretching is to increase the dynamic range of the gray levels in the image being processed.

Figure 3.10(a) shows a typical transformation used for contrast stretching. The locations of points (r_1, s_1) and (r_2, s_2) control the shape of the transformation



a b
c d

FIGURE 3.10 Contrast stretching. (a) Form of transformation function. (b) A low-contrast image. (c) Result of contrast stretching. (d) Result of thresholding. (Original image courtesy of Dr. Roger Heady, Research School of Biological Sciences, Australian National University, Canberra, Australia.)

function. If $r_1 = s_1$ and $r_2 = s_2$, the transformation is a linear function that produces no changes in gray levels. If $r_1 = r_2$, $s_1 = 0$ and $s_2 = L - 1$, the transformation becomes a *thresholding function* that creates a binary image, as illustrated in Fig. 3.2(b). Intermediate values of (r_1, s_1) and (r_2, s_2) produce various degrees of spread in the gray levels of the output image, thus affecting its contrast. In general, $r_1 \leq r_2$ and $s_1 \leq s_2$ is assumed so that the function is single valued and monotonically increasing. This condition preserves the order of gray levels, thus preventing the creation of intensity artifacts in the processed image.

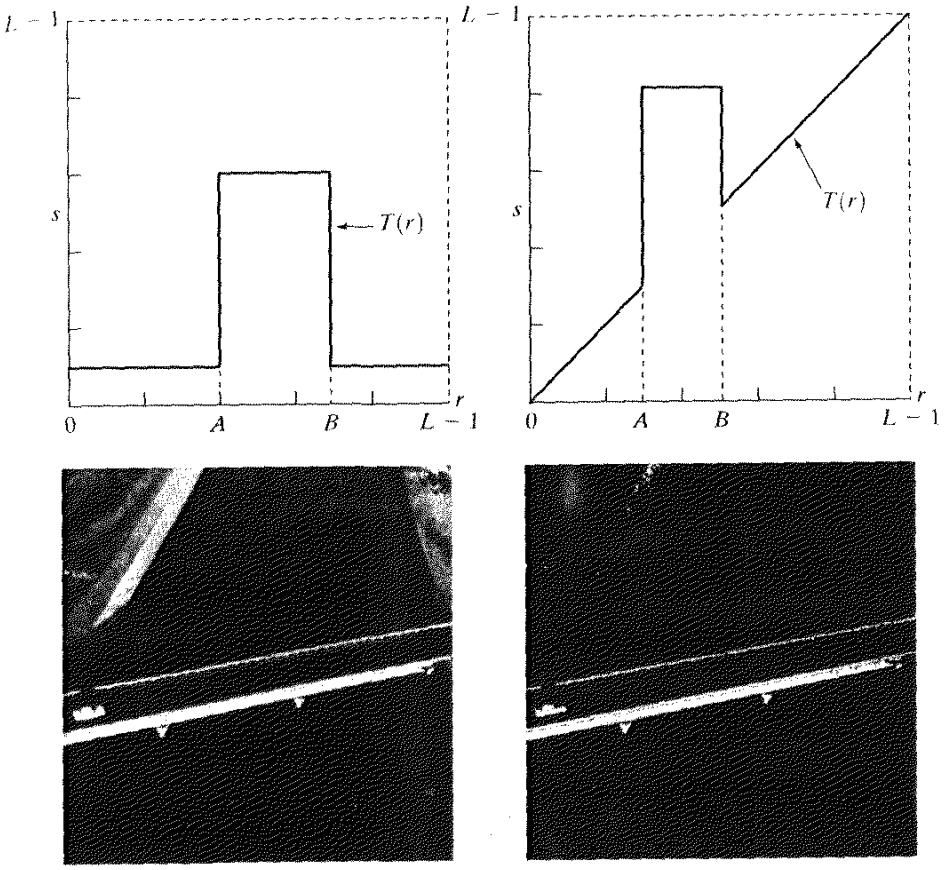
Figure 3.10(b) shows an 8-bit image with low contrast. Fig. 3.10(c) shows the result of contrast stretching, obtained by setting $(r_1, s_1) = (r_{\min}, 0)$ and $(r_2, s_2) = (r_{\max}, L - 1)$ where $r_{\min}$ and $r_{\max}$ denote the minimum and maximum gray levels in the image, respectively. Thus, the transformation function stretched the levels linearly from their original range to the full range $[0, L - 1]$. Finally, Fig. 3.10(d) shows the result of using the thresholding function defined previously, with $r_1 = r_2 = m$, the mean gray level in the image. The original image on which these results are based is a scanning electron microscope image of pollen, magnified approximately 700 times.

Gray-level slicing

Highlighting a specific range of gray levels in an image often is desired. Applications include enhancing features such as masses of water in satellite imagery and enhancing flaws in X-ray images. There are several ways of doing level slicing, but most of them are variations of two basic themes. One approach is to display a high value for all gray levels in the range of interest and a low value for all other gray levels. This transformation, shown in Fig. 3.11(a), produces a binary image. The second approach, based on the transformation shown in Fig. 3.11(b), brightens the desired range of gray levels but preserves the background and gray-level tonalities in the image. Figure 3.11(c) shows a gray-scale image, and Fig. 3.11(d) shows the result of using the transformation in Fig. 3.11(a). Variations of the two transformations shown in Fig. 3.11 are easy to formulate.

Bit-plane slicing

Instead of highlighting gray-level ranges, highlighting the contribution made to total image appearance by specific bits might be desired. Suppose that each pixel in an image is represented by 8 bits. Imagine that the image is composed of eight 1-bit planes, ranging from bit-plane 0 for the least significant bit to bit-plane 7 for the most significant bit. In terms of 8-bit bytes, plane 0 contains all the lowest order bits in the bytes comprising the pixels in the image and plane 7 contains all the high-order bits. Figure 3.12 illustrates these ideas, and Fig. 3.14 shows the various bit planes for the image shown in Fig. 3.13. Note that the higher-order bits (especially the top four) contain the majority of the visually significant data. The other bit planes contribute to more subtle details in the image. Separating a digital image into its bit planes is useful for analyzing the relative importance played by each bit of the image, a process that aids in determining the adequacy of the number of bits used to quantize each pixel. Also, this type of decomposition is useful for image compression, as discussed in Chapter 8.



a b
c d

FIGURE 3.11
(a) This transformation highlights range $[A, B]$ of gray levels and reduces all others to a constant level. (b) This transformation highlights range $[A, B]$ but preserves all other levels. (c) An image. (d) Result of using the transformation in (a).

In terms of bit-plane extraction for an 8-bit image, it is not difficult to show that the (binary) image for bit-plane 7 can be obtained by processing the input image with a thresholding gray-level transformation function that (1) maps all levels in the image between 0 and 127 to one level (for example, 0); and (2) maps all levels between 129 and 255 to another (for example, 255). The binary image for bit-plane 7 in Fig. 3.14 was obtained in just this manner. It is left as an exercise (Problem 3.3) to obtain the gray-level transformation functions that would yield the other bit planes.

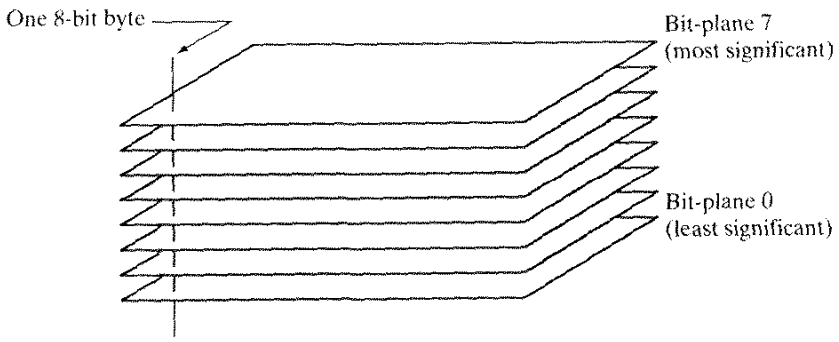


FIGURE 3.12
Bit-plane representation of an 8-bit image.

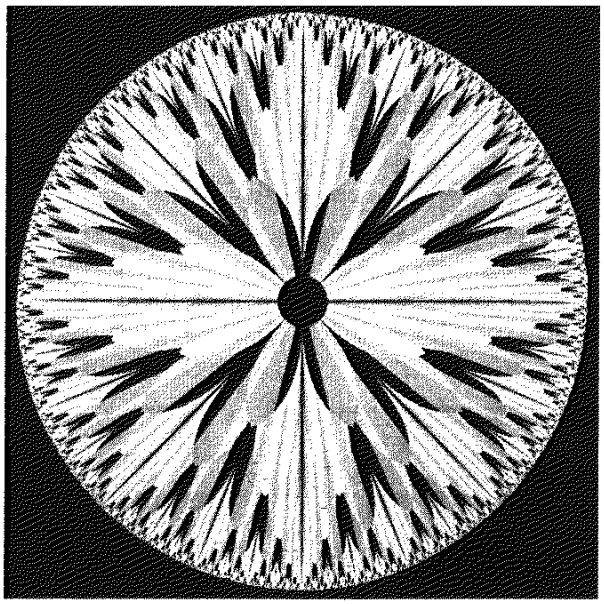


FIGURE 3.13 An 8-bit fractal image. (A fractal is an image generated from mathematical expressions). (Courtesy of Ms. Melissa D. Binde, Swarthmore College, Swarthmore, PA.)

See inside front cover
Consult the book web site for a review of basic probability theory.

3.3 Histogram Processing

The histogram of a digital image with gray levels in the range $[0, L-1]$ is a discrete function $h(r_k) = n_k$, where r_k is the kth gray level and n_k is the number of pixels in the image having gray level r_k. It is common practice to normalize a histogram by dividing each of its values by the total number of pixels in the image, denoted by n. Thus, a normalized histogram is given by $p(r_k) = n_k/n$, for $k = 0, 1, \ldots, L-1$. Loosely speaking, $p(r_k)$ gives an estimate of the probability of occurrence of gray level r_k. Note that the sum of all components of a normalized histogram is equal to 1.

Histograms are the basis for numerous spatial domain processing techniques. Histogram manipulation can be used effectively for image enhancement, as shown in this section. In addition to providing useful image statistics, we shall see in subsequent chapters that the information inherent in histograms also is quite useful in other image processing applications, such as image compression and segmentation. Histograms are simple to calculate in software and also lend themselves to economic hardware implementations, thus making them a popular tool for real-time image processing.

As an introduction to the role of histogram processing in image enhancement, consider Fig. 3.15, which is the pollen image of Fig. 3.10 shown in four basic gray-level characteristics: dark, light, low contrast, and high contrast. The right side of the figure shows the histograms corresponding to these images. The horizontal axis of each histogram plot corresponds to gray level values, r_k. The vertical axis corresponds to values of $h(r_k) = n_k$ or $p(r_k) = n_k/n$ if the values are normalized. Thus, as indicated previously, these histogram plots are simply plots of $h(r_k) = n_k$ versus r_k or $p(r_k) = n_k/n$ versus r_k.

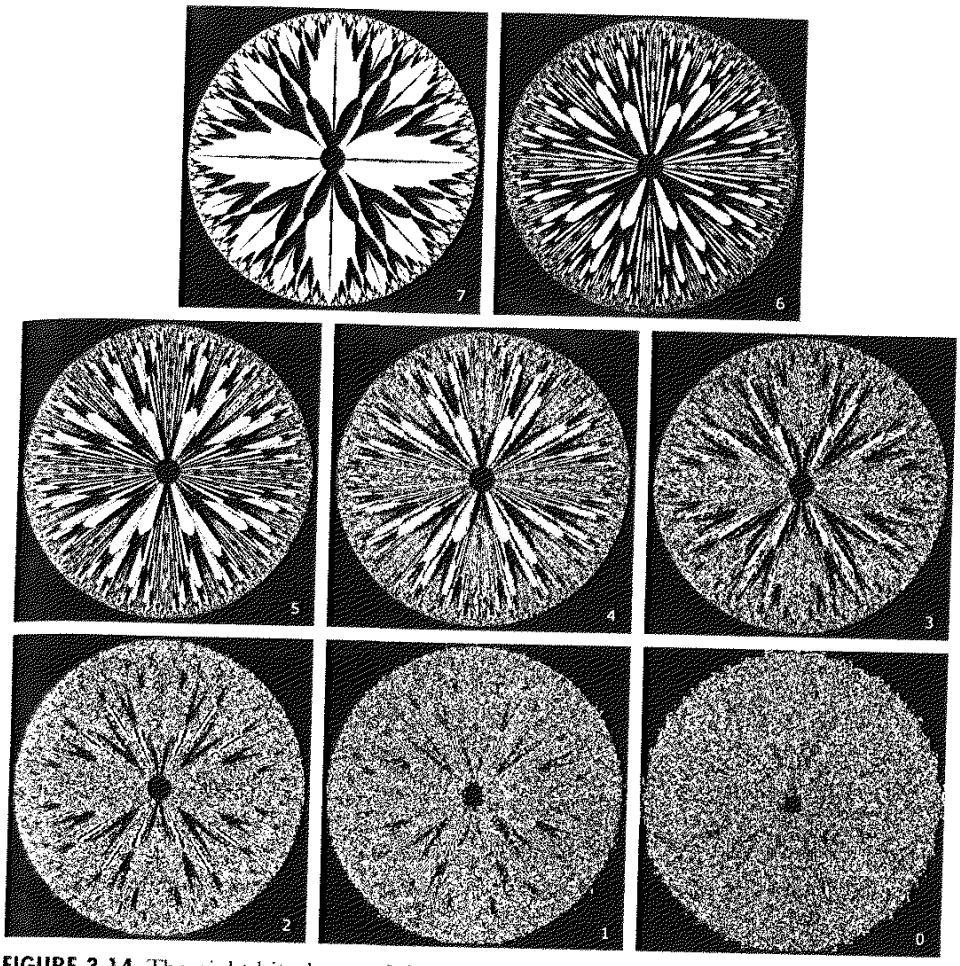
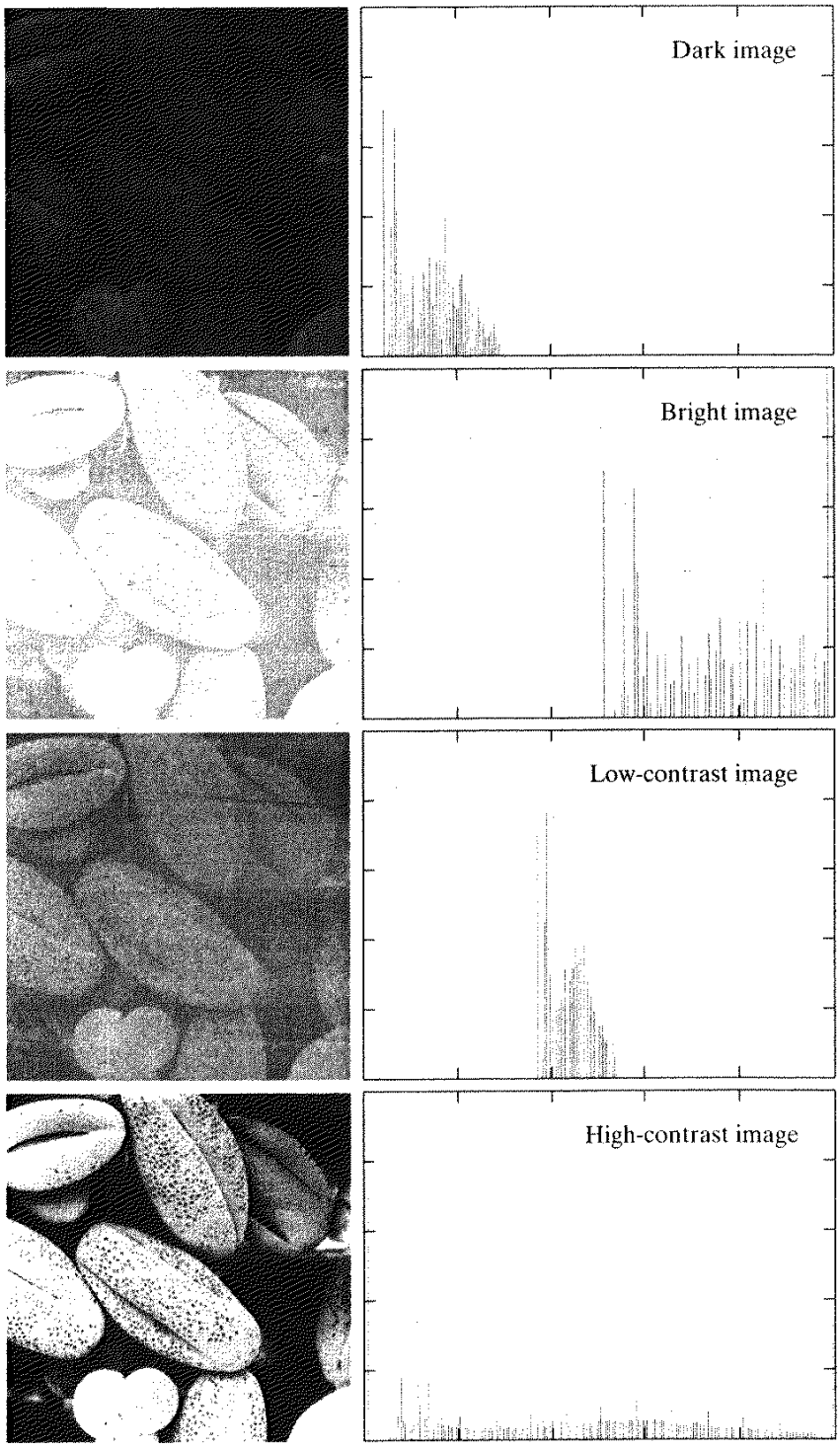


FIGURE 3.14 The eight bit planes of the image in Fig. 3.13. The number at the bottom, right of each image identifies the bit plane.

We note in the dark image that the components of the histogram are concentrated on the low (dark) side of the gray scale. Similarly, the components of the histogram of the bright image are biased toward the high side of the gray scale. An image with low contrast has a histogram that will be narrow and will be centered toward the middle of the gray scale. For a monochrome image this implies a dull, washed-out gray look. Finally, we see that the components of the histogram in the high-contrast image cover a broad range of the gray scale and, further, that the distribution of pixels is not too far from uniform, with very few vertical lines being much higher than the others. Intuitively, it is reasonable to conclude that an image whose pixels tend to occupy the entire range of possible gray levels and, in addition, tend to be distributed uniformly, will have an appearance of high contrast and will exhibit a large variety of gray tones. The net effect will be an image that shows a great deal of gray-level detail and has high dynamic range. It will be shown shortly that it is possible to develop a transformation function that can automatically achieve this effect, based only on information available in the histogram of the input image.



a b

FIGURE 3.15 Four basic image types: dark, light, low contrast, high contrast, and their cor responding histograms. (Original image courtesy of Dr. Roger Heady, Research Schoc of Biological Sciences, Australian National University, Canberra, Australia.)

Histogram Equalization

Consider for a moment continuous functions, and let the variable r represent the gray levels of the image to be enhanced. In the initial part of our discussion we assume that r has been normalized to the interval $[0, 1]$, with $r = 0$ representing black and $r = 1$ representing white. Later, we consider a discrete formulation and allow pixel values to be in the interval $[0, L - 1]$.

For any r satisfying the aforementioned conditions, we focus attention on transformations of the form

$$s = T(r) \qquad 0 \leq r \leq 1 \tag{3.3-1}$$

that produce a level s for every pixel value r in the original image. For reasons that will become obvious shortly, we assume that the transformation function $T(r)$ satisfies the following conditions:

(a) $T(r)$ is single-valued and monotonically increasing in the interval $0 \leq r \leq 1$; and

(b) $0 \leq T(r) \leq 1$ for $0 \leq r \leq 1$.

The requirement in (a) that $T(r)$ be single valued is needed to guarantee that the inverse transformation will exist, and the monotonicity condition preserves the increasing order from black to white in the output image. A transformation function that is not monotonically increasing could result in at least a section of the intensity range being inverted, thus producing some inverted gray levels in the output image. While this may be a desirable effect in some cases, that is not what we are after in the present discussion. Finally, condition (b) guarantees that the output gray levels will be in the same range as the input levels. Figure 3.16 gives an example of a transformation function that satisfies these two conditions. The inverse transformation from s back to r is denoted

$$r = T^{-1}(s) \qquad 0 \leq s \leq 1. \tag{3.3-2}$$

It can be shown by example (Problem 3.8) that even if $T(r)$ satisfies conditions (a) and (b), it is possible that the corresponding inverse $T^{-1}(s)$ may fail to be single valued.

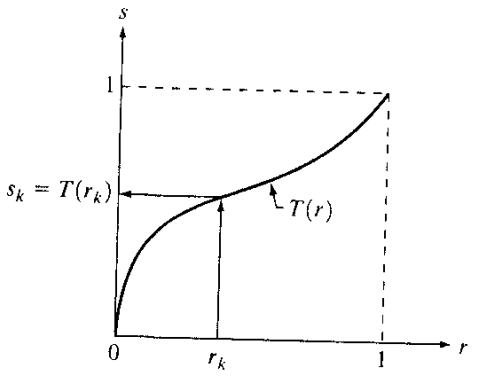


FIGURE 3.16 A gray-level transformation function that is both single valued and monotonically increasing.

The gray levels in an image may be viewed as random variables in the interval [0, 1]. One of the most fundamental descriptors of a random variable is its probability density function (PDF). Let $p_r(r)$ and $p_s(s)$ denote the probability density functions of random variables r and s, respectively, where the subscripts on p are used to denote that p_r and p_s are different functions. A basic result from an elementary probability theory is that, if $p_r(r)$ and $T(r)$ are known and $T^{-1}(s)$ satisfies condition (a), then the probability density function $p_s(s)$ of the transformed variable s can be obtained using a rather simple formula:

$$p_s(s) = p_r(r)\left|\frac{dr}{ds}\right|. \tag{3.3-3}$$

Thus, the probability density function of the transformed variable, s, is determined by the gray-level PDF of the input image and by the chosen transformation function.

A transformation function of particular importance in image processing has the form

$$s = T(r) = \int_0^r p_r(w)\,dw \tag{3.3-4}$$

where w is a dummy variable of integration. The right side of Eq. (3.3-4) is recognized as the cumulative distribution function (CDF) of random variable r. Since probability density functions are always positive, and recalling that the integral of a function is the area under the function, it follows that this transformation function is single valued and monotonically increasing, and, therefore, satisfies condition (a). Similarly, the integral of a probability density function for variables in the range [0, 1] also is in the range [0, 1], so condition (b) is satisfied as well.

Given transformation function $T(r)$, we find $p_s(s)$ by applying Eq. (3.3-3). We know from basic calculus (Leibniz's rule) that the derivative of a definite integral with respect to its upper limit is simply the integrand evaluated at that limit. In other words,

$$\begin{aligned}\frac{ds}{dr} &= \frac{dT(r)}{dr} \\ &= \frac{d}{dr}\left[\int_0^r p_r(w)\,dw\right] \\ &= p_r(r).\end{aligned} \tag{3.3-5}$$

Substituting this result for dr/ds into Eq. (3.3-3), and keeping in mind that all probability values are positive, yields

$$\begin{aligned}p_s(s) &= p_r(r)\left|\frac{dr}{ds}\right| \\ &= p_r(r)\left|\frac{1}{p_r(r)}\right| \\ &= 1 \qquad 0 \le s \le 1.\end{aligned} \tag{3.3-6}$$

Because $p_s(s)$ is a probability density function, it follows that it must be zero outside the interval [0, 1] in this case because its integral over all values of s must equal 1. We recognize the form of $p_s(s)$ given in Eq. (3.3-6) as a *uniform* probability density function. Simply stated, we have demonstrated that performing the transformation function given in Eq. (3.3-4) yields a random variable s characterized by a uniform probability density function. It is important to note from Eq. (3.3-4) that $T(r)$ depends on $p_r(r)$, but, as indicated by Eq. (3.3-6), the resulting $p_s(s)$ *always* is uniform, *independent* of the form of $p_r(r)$.

For discrete values we deal with probabilities and summations instead of probability density functions and integrals. The probability of occurrence of gray level r_k in an image is approximated by

$$p_r(r_k) = \frac{n_k}{n} \qquad k = 0, 1, 2, \ldots, L - 1 \tag{3.3-7}$$

where, as noted at the beginning of this section, n is the total number of pixels in the image, n_k is the number of pixels that have gray level r_k, and L is the total number of possible gray levels in the image. The discrete version of the transformation function given in Eq. (3.3-4) is

$$\begin{aligned} s_k = T(r_k) &= \sum_{j=0}^{k} p_r(r_j) \\ &= \sum_{j=0}^{k} \frac{n_j}{n} \qquad k = 0, 1, 2, \ldots, L - 1. \end{aligned} \tag{3.3-8}$$

Thus, a processed (output) image is obtained by mapping each pixel with level r_k in the input image into a corresponding pixel with level s_k in the output image via Eq. (3.3-8). As indicated earlier, a plot of $p_r(r_k)$ versus r_k is called a *histogram*. The transformation (mapping) given in Eq. (3.3-8) is called *histogram equalization* or *histogram linearization*. It is not difficult to show (Problem 3.9) that the transformation in Eq. (3.3-8) satisfies conditions (a) and (b) stated previously in this section.

Unlike its continuous counterpart, it cannot be proved in general that this discrete transformation will produce the discrete equivalent of a uniform probability density function, which would be a uniform histogram. However, as will be seen shortly, use of Eq. (3.3-8) does have the general tendency of spreading the histogram of the input image so that the levels of the histogram-equalized image will span a fuller range of the gray scale.

We discussed earlier in this section the many advantages of having gray-level values that cover the entire gray scale. In addition to producing gray levels that have this tendency, the method just derived has the additional advantage that it is fully "automatic." In other words, given an image, the process of histogram equalization consists simply of implementing Eq. (3.3-8), which is based on information that can be extracted directly from the given image, without the need for further parameter specifications. We note also the simplicity of the computations that would be required to implement the technique.

The inverse transformation from s back to r is denoted by

$$r_k = T^{-1}(s_k) \qquad k = 0, 1, 2, \ldots, L - 1 \tag{3.3-9}$$

It can be shown (Problem 3.9) that the inverse transformation in Eq. (3.3-9) satisfies conditions (a) and (b) stated previously in this section only if none of the levels, $r_k, k = 0, 1, 2, \ldots, L - 1$, are missing from the input image. Although the inverse transformation is not used in histogram equalization, it plays a central role in the histogram-matching scheme developed in the next section. We also discuss in that section details of how to implement histogram processing techniques.

EXAMPLE 3.3: Histogram equalization.

Figure 3.17(a) shows the four images from Fig. 3.15, and Fig. 3.17(b) shows the result of performing histogram equalization on each of these images. The first three results (top to bottom) show significant improvement. As expected, histogram equalization did not produce a significant visual difference in the fourth image because the histogram of this image already spans the full spectrum of the gray scale. The transformation functions used to generate the images in Fig. 3.17(b) are shown in Fig. 3.18. These functions were generated from the histograms of the original images [see Fig. 3.15(b)] using Eq. (3.3-8). Note that transformation (4) has a basic linear shape, again indicating that the gray levels in the fourth input image are nearly uniformly distributed. As was just noted, we would expect histogram equalization in this case to have negligible effect on the appearance of the image.

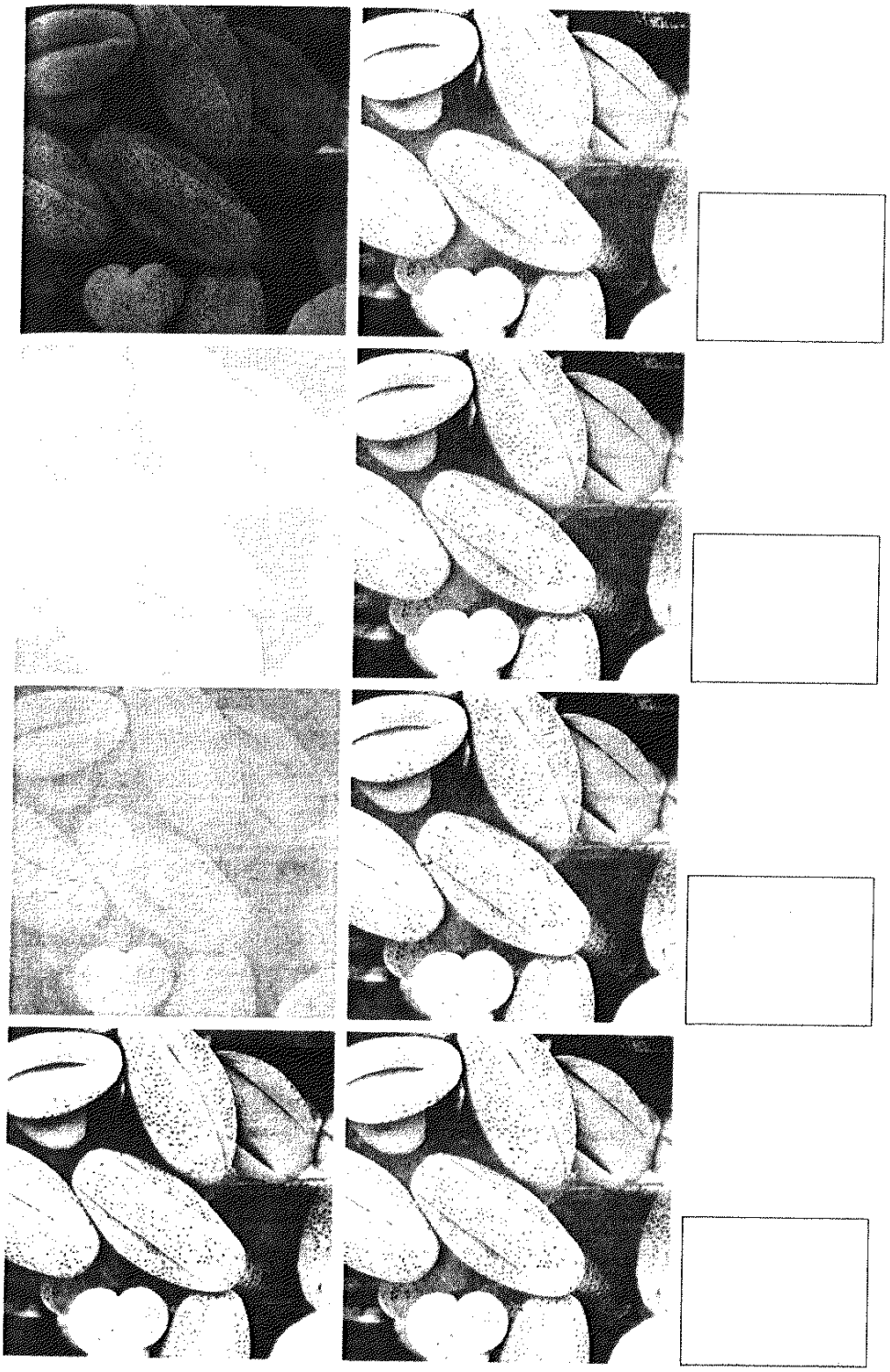
The histograms of the equalized images are shown in Fig. 3.17(c). It is of interest to note that, while all these histograms are different, the histogram-equalized images themselves are visually very similar. This is not unexpected because the difference between the images in the left column is simply one of contrast, not of content. In other words, since the images have the same content, the increase in contrast resulting from histogram equalization was enough to render any gray-level differences in the resulting images visually indistinguishable. Given the significant contrast differences of the images in the left column, this example illustrates the power of histogram equalization as an adaptive enhancement tool.

3.3.2 Histogram Matching (Specification)

As indicated in the preceding discussion, histogram equalization automatically determines a transformation function that seeks to produce an output image that has a uniform histogram. When automatic enhancement is desired, this is a good approach because the results from this technique are predictable and the method is simple to implement. We show in this section that there are applications in which attempting to base enhancement on a uniform histogram is not the best approach. In particular, it is useful sometimes to be able to specify the shape of the histogram that we wish the processed image to have. The method used to generate a processed image that has a specified histogram is called *histogram matching* or *histogram specification*.

Development of the method

Let us return for a moment to continuous gray levels r and z (considered continuous random variables), and let $p_r(r)$ and $p_z(z)$ denote their corresponding continuous probability density functions. In this notation, r and z denote



a b c

FIGURE 3.17 (a) Images from Fig. 3.15. (b) Results of histogram equalization. (c) Corresponding histograms.

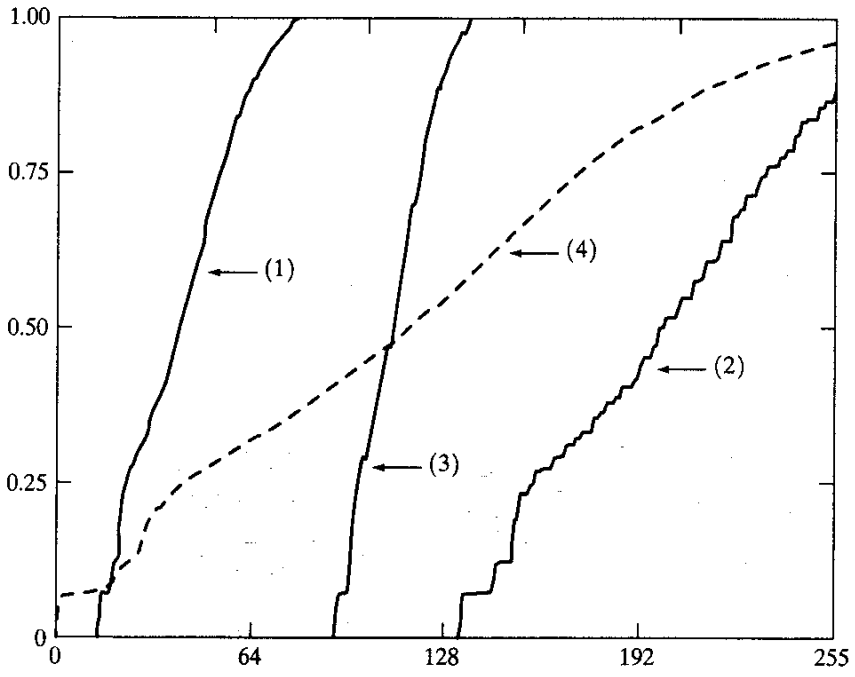


FIGURE 3.18 Transformation functions (1) through (4) were obtained from the histograms of the images in Fig.3.17(a), using Eq. (3.3-8).

the gray levels of the input and output (processed) images, respectively. We can estimate $p_r(r)$ from the given input image, while $p_z(z)$ is the *specified* probability density function that we wish the output image to have.

Let s be a random variable with the property

$$s = T(r) = \int_0^r p_r(w)\, dw \tag{3.3-10}$$

where w is a dummy variable of integration. We recognize this expression as the continuos version of histogram equalization given in Eq. (3.3-4). Suppose next that we define a random variable z with the property

$$G(z) = \int_0^z p_z(t)\, dt = s \tag{3.3-11}$$

where t is a dummy variable of integration. It then follows from these two equations that $G(z) = T(r)$ and, therefore, that z must satisfy the condition

$$z = G^{-1}(s) = G^{-1}[T(r)]. \tag{3.3-12}$$

The transformation $T(r)$ can be obtained from Eq. (3.3-10) once $p_r(r)$ has been estimated from the input image. Similarly, the transformation function $G(z)$ can be obtained using Eq. (3.3-11) because $p_z(z)$ is given.

Assuming that G^{-1} exists and that it satisfies conditions (a) and (b) in the previous section, Eqs. (3.3-10) through (3.3-12) show that an image with a specified probability density function can be obtained from an input image by using the following procedure: (1) Obtain the transformation function $T(r)$ using Eq. (3.3-10). (2) Use Eq. (3.3-11) to obtain the transformation function $G(z)$. (3) Obtain the inverse transformation function G^{-1}. (4) Obtain the output image

by applying Eq. (3.3-12) to all the pixels in the input image. The result of this procedure will be an image whose gray levels, z, have the specified probability density function $p_z(z)$.

Although the procedure just described is straightforward in principle, it is seldom possible in practice to obtain analytical expressions for $T(r)$ and for G^{-1}. Fortunately, this problem is simplified considerably in the case of discrete values. The price we pay is the same as in histogram equalization, where only an approximation to the desired histogram is achievable. In spite of this, however, some very useful results can be obtained even with crude approximations.

The discrete formulation of Eq. (3.3-10) is given by Eq. (3.3-8), which we repeat here for convenience:

$$\begin{aligned} s_k = T(r_k) &= \sum_{j=0}^{k} p_r(r_j) \\ &= \sum_{j=0}^{k} \frac{n_j}{n} \qquad k = 0, 1, 2, \ldots, L-1 \end{aligned} \tag{3.3-13}$$

where n is the total number of pixels in the image, n_j is the number of pixels with gray level r_j, and L is the number of discrete gray levels. Similarly, the discrete formulation of Eq. (3.3-11) is obtained from the given histogram $p_z(z_i), i = 0, 1, 2, \ldots, L-1$, and has the form

$$v_k = G(z_k) = \sum_{i=0}^{k} p_z(z_i) = s_k \qquad k = 0, 1, 2, \ldots, L-1. \tag{3.3-14}$$

As in the continuous case, we are seeking values of z that satisfy this equation. The variable v_k was added here for clarity in the discussion that follows. Finally, the discrete version of Eq. (3.3-12) is given by

$$z_k = G^{-1}[T(r_k)] \qquad k = 0, 1, 2, \ldots, L-1 \tag{3.3-15}$$

or, from Eq. (3.3-13),

$$z_k = G^{-1}(s_k) \qquad k = 0, 1, 2, \ldots, L-1. \tag{3.3-16}$$

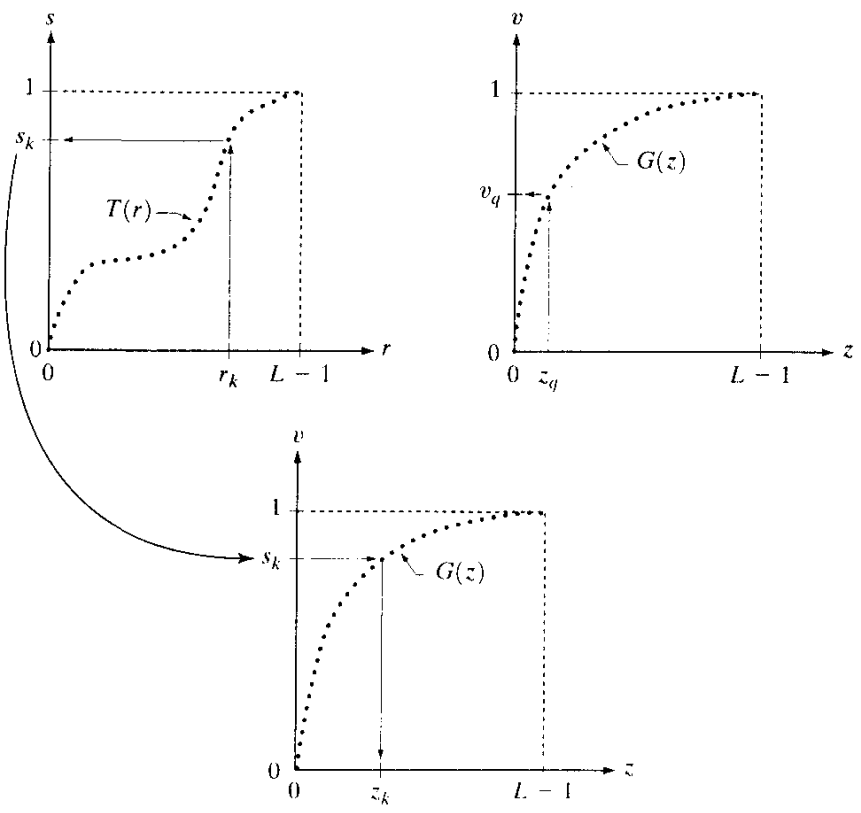
Equations (3.3-13) through (3.3-16) are the foundation for implementing histogram matching for digital images. Equation (3.3-13) is a mapping from the levels in the original image into corresponding levels s_k based on the histogram of the original image, which we compute from the pixels in the image. Equation (3.3-14) computes a transformation function G from the given histogram $p_z(z)$. Finally, Eq. (3.3-15) or its equivalent, Eq. (3.3-16), gives us (an approximation of) the desired levels of the image with that histogram. The first two equations can be implemented easily because all the quantities are known. Implementation of Eq. (3.3-16) is straightforward, but requires additional explanation.

Implementation

We start by noting the following: (1) Each set of gray levels $\{r_j\}$, $\{s_j\}$, and $\{z_j\}$, $j = 0, 1, 2, \ldots, L-1$, is a one-dimensional array of dimension $L \times 1$. (2) All mappings from r to s and from s to z are simple table lookups between a given

pixel value and these arrays. (3) Each of the elements of these arrays, for example, s_k, contains two important pieces of information: The subscript k denotes the location of the element in the array, and s denotes the value at that location. (4) We need to be concerned only with integer pixel values. For example, in the case of an 8-bit image, $L = 256$ and the elements of each of the arrays just mentioned are integers between 0 and 255. This implies that we now work with gray level values in the interval $[0, L - 1]$ instead of the normalized interval $[0, 1]$ that we used before to simplify the development of histogram processing techniques.

In order to see how histogram matching actually can be implemented, consider Fig. 3.19(a), ignoring for a moment the connection shown between this figure and Fig. 3.19(c). Figure 3.19(a) shows a hypothetical discrete transformation function $s = T(r)$ obtained from a given image. The first gray level in the image, r_1, maps to s_1; the second gray level, r_2, maps to s_2; the kth level r_k maps to s_k; and so on (the important point here is the *ordered* correspondence between these values). Each value s_j in the array is precomputed using Eq. (3.3-13), so the process of mapping simply uses the actual value of a pixel as an index in an array to determine the corresponding value of s. This process is particularly easy because we are dealing with integers. For example, the s mapping for an 8-bit pixel with value 127 would be found in the 128th position in array $\{s_j\}$ (recall that we start at 0) out of the possible 256 positions. If we stopped here and mapped the value of each pixel of an input image by the



a b
c

FIGURE 3.19
(a) Graphical interpretation of mapping from r_k to s_k via $T(r)$. (b) Mapping of z_q to its corresponding value v_q via $G(z)$. (c) Inverse mapping from s_k to its corresponding value of z_k.

method just described, the output would be a histogram-equalized image, according to Eq. (3.3-8).

In order to implement histogram matching we have to go one step further. Figure 3.19(b) is a hypothetical transformation function G obtained from a given histogram $p_z(z)$ by using Eq. (3.3-14). For any z_q, this transformation function yields a corresponding value v_q. This mapping is shown by the arrows in Fig. 3.19(b). Conversely, given any value v_q, we would find the corresponding value z_q from G^{-1}. In terms of the figure, all this means graphically is that we would reverse the direction of the arrows to map v_q into its corresponding z_q. However, we know from the definition in Eq. (3.3-14) that $v = s$ for corresponding subscripts, so we can use exactly this process to find the z_k corresponding to any value s_k that we computed previously from the equation $s_k = T(r_k)$. This idea is shown in Fig. 3.19(c).

Since we really do not have the z's (recall that finding these values is precisely the objective of histogram matching), we must resort to some sort of iterative scheme to find z from s. The fact that we are dealing with integers makes this a particularly simple process. Basically, because $v_k = s_k$, we have from Eq. (3.3-14) that the z's for which we are looking must satisfy the equation $G(z_k) = s_k$, or $(G(z_k) - s_k) = 0$. Thus, all we have to do to find the value of z_k corresponding to s_k is to iterate on values of z such that this equation is satisfied for $k = 0, 1, 2, \ldots, L - 1$. This is the same thing as Eq. (3.3-16), except that we do not have to find the inverse of G because we are going to iterate on z. Since we are dealing with integers, the closest we can get to satisfying the equation $(G(z_k) - s_k) = 0$ is to let $z_k = \hat{z}$ for each value of k, where $\hat{z}$ is the *smallest* integer in the interval $[0, L - 1]$ such that

$$(G(\hat{z}) - s_k) \geq 0 \qquad k = 0, 1, 2, \ldots, L - 1. \tag{3.3-17}$$

Given a value s_k, all this means conceptually in terms of Fig. 3.19(c) is that we would start with $\hat{z} = 0$ and increase it in integer steps until Eq. (3.3-17) is satisfied, at which point we let $z_k = \hat{z}$. Repeating this process for all values of k would yield all the required mappings from s to z, which constitutes the implementation of Eq. (3.3-16). In practice, we would not have to start with $\hat{z} = 0$ each time because the values of s_k are known to increase monotonically. Thus, for $k = k + 1$, we would start with $\hat{z} = z_k$ and increment in integer values from there.

The procedure we have just developed for histogram matching may be summarized as follows:

1. Obtain the histogram of the given image.
2. Use Eq. (3.3-13) to precompute a mapped level s_k for each level r_k.
3. Obtain the transformation function G from the given $p_z(z)$ using Eq. (3.3-14).
4. Precompute z_k for each value of s_k using the iterative scheme defined in connection with Eq. (3.3-17).
5. For each pixel in the original image, if the value of that pixel is r_k, map this value to its corresponding level s_k; then map level s_k into the final level z_k. Use the precomputed values from Steps (2) and (4) for these mappings.

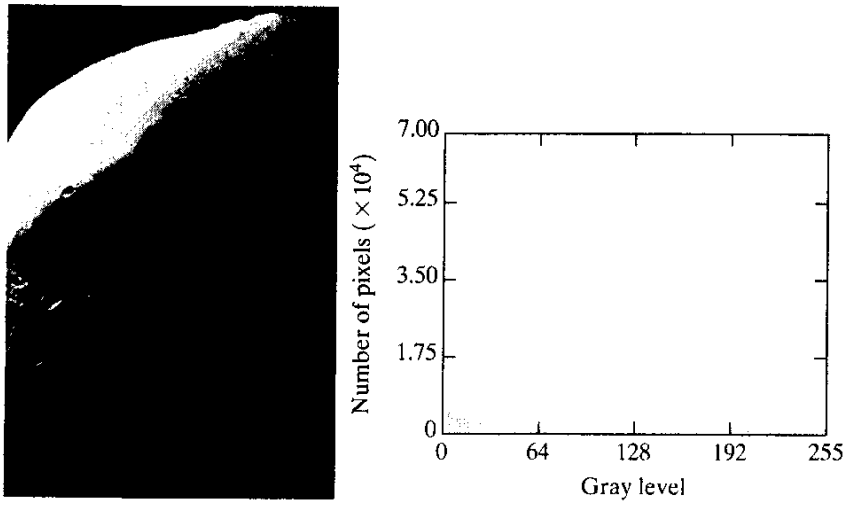
Note that Step (5) implements two mappings for each pixel in the image being processed. The first mapping is nothing more than histogram equalization. If the histogram-equalized image is not required, it obviously would be beneficial to combine both transformations into one in order to save an intermediate step.

Finally, we note that, even in the discrete case, we need to be concerned about G^{-1} satisfying conditions (a) and (b) of the previous section. It is not difficult to show (Problem 3.9) that the only way to guarantee that G^{-1} be single valued and monotonic is to require that G be strictly monotonic (i.e., always increasing), which means simply that none of the values of the specified histogram $p_z(z_i)$ in Eq. (3.3-14) can be zero.

EXAMPLE 3.4: Comparison between histogram equalization and histogram matching.

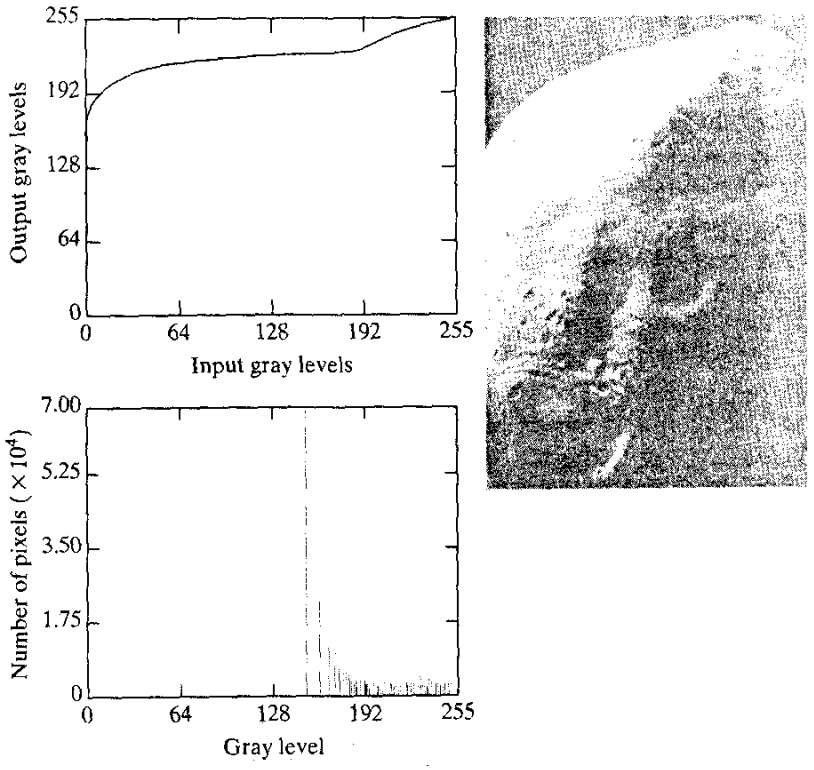
■ Figure 3.20(a) shows an image of the Mars moon, Phobos, taken by NASA's *Mars Global Surveyor*. Figure 3.20(b) shows the histogram of Fig. 3.20(a). The image is dominated by large, dark areas, resulting in a histogram characterized by a large concentration of pixels in the dark end of the gray scale. At first glance, one might conclude that histogram equalization would be a good approach to enhance this image, so that details in the dark areas become more visible. It is demonstrated in the following discussion that this is not so.

Figure 3.21(a) shows the histogram equalization transformation [Eq. (3.3-8) or (3.3-13)] obtained from the histogram shown in Fig. 3.20(b). The most relevant characteristic of this transformation function is how fast it rises from gray level 0 to a level near 190. This is caused by the large concentration of pixels in the input histogram having levels very near 0. When this transformation is applied to the levels of the input image to obtain a histogram-equalized result, the net effect is to map a very narrow interval of dark pixels into the upper end of the gray scale of the output image. Because numerous pixels in the input image have levels precisely in this interval, we would expect the result to be an



a b

FIGURE 3.20 (a) Image of the Mars moon Phobos taken by NASA's *Mars Global Surveyor*. (b) Histogram. (Original image courtesy of NASA.)



a b
c

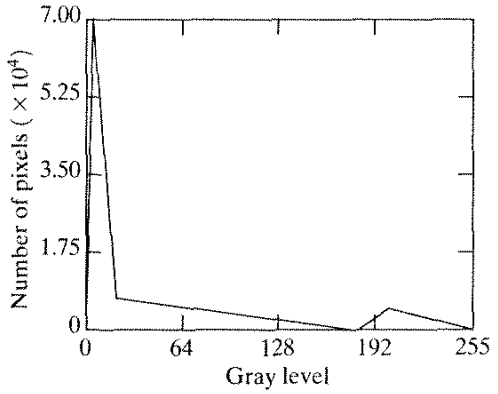
FIGURE 3.21
(a) Transformation function for histogram equalization. (b) Histogram-equalized image (note the washed-out appearance). (c) Histogram of (b).

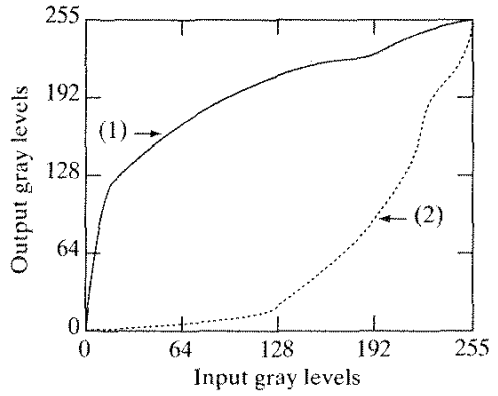
image with a light, washed-out appearance. As shown in Fig. 3.21(b), this is indeed the case. The histogram of this image is shown in Fig. 3.21(c). Note how all the gray levels are biased toward the upper one-half of the gray scale.

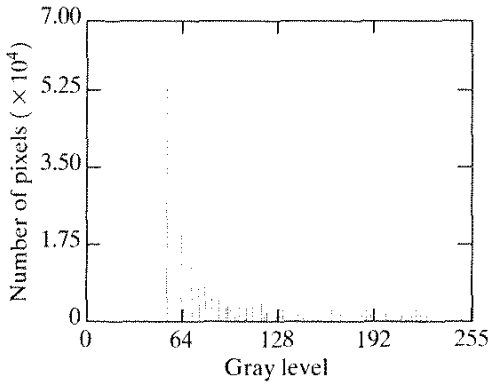
Since the problem with the transformation function in Fig. 3.21(a) was caused by a large concentration of pixels in the original image with levels near 0, a reasonable approach is to modify the histogram of that image so that it does not have this property. Figure 3.22(a) shows a *manually specified* function that preserves the general shape of the original histogram, but has a smoother transition of levels in the dark region of the gray scale. Sampling this function into 256 equally spaced discrete values produced the desired specified histogram. The transformation function $G(z)$ obtained from this histogram using Eq. (3.3-14) is labeled transformation (1) in Fig. 3.22(b). Similarly, the inverse transformation $G^{-1}(s)$ from Eq. (3.3-16) [obtained using the iterative technique discussed in connection with Eq. (3.3-17)] is labeled transformation (2) in Fig. 3.22(b). The enhanced image in Fig. 3.22(c) was obtained by applying transformation (2) to the pixels of the histogram-equalized image in Fig. 3.21(b). The improvement of the histogram-specified image over the result obtained by histogram equalization is evident by comparing these two images. It is of interest to note that a rather modest change in the original histogram was all that was required to obtain a significant improvement in enhancement. The histogram of Fig. 3.22(c) is shown in Fig. 3.22(d). The most distinguishing feature of this histogram is how its low end has shifted right toward the lighter region of the gray scale, as desired. ■

a c
b
d

FIGURE 3.22
(a) Specified histogram.
(b) Curve (1) is from Eq. (3.3-14), using the histogram in (a); curve (2) was obtained using the iterative procedure in Eq. (3.3-17).
(c) Enhanced image using mappings from curve (2).
(d) Histogram of (c).







Although it probably is obvious by now, we emphasize before leaving this section that histogram specification is, for the most part, a trial-and-error process. One can use guidelines learned from the problem at hand, just as we did in the preceding example. At times, there may be cases in which it is possible to formulate what an "average" histogram should look like and use that as the specified histogram. In cases such as these, histogram specification becomes a straightforward process. In general, however, there are no rules for specifying histograms, and one must resort to analysis on a case-by-case basis for any given enhancement task.

Local Enhancement

The histogram processing methods discussed in the previous two sections are *global*, in the sense that pixels are modified by a transformation function based on the gray-level content of an entire image. Although this global approach is suitable for overall enhancement, there are cases in which it is necessary to enhance details over small areas in an image. The number of pixels in these areas may have negligible influence on the computation of a global transformation whose shape does not necessarily guarantee the desired local enhancement. The solution is to devise transformation functions based on the gray-level distribution—or other properties—in the neighborhood of every pixel in the image. Although processing methods based on neighborhoods are the topic of Section 3.5, we discuss local histogram processing here for the sake of clarity and continuity. The reader will have no difficulty in following the discussion.

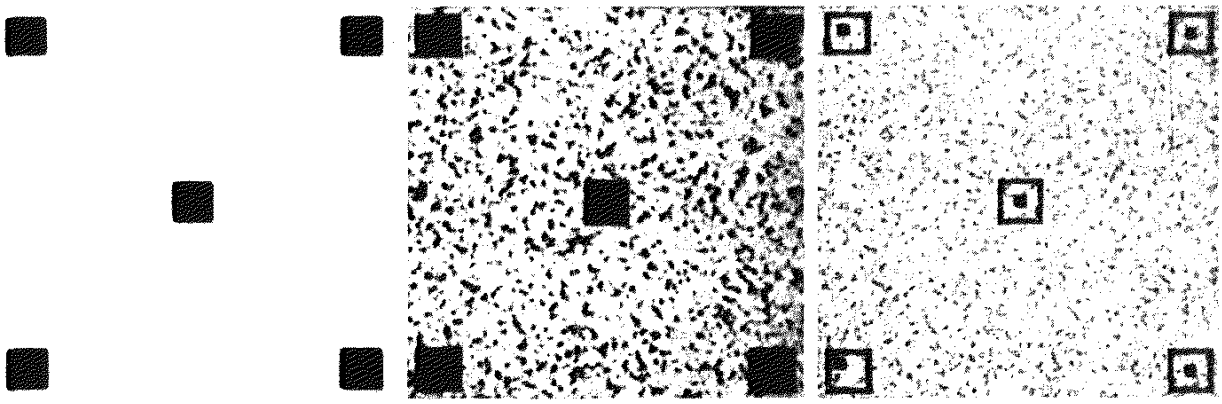
The histogram processing techniques previously described are easily adaptable to local enhancement. The procedure is to define a square or rectangular neighborhood and move the center of this area from pixel to pixel. At each location, the histogram of the points in the neighborhood is computed and either a histogram equalization or histogram specification transformation function is obtained. This function is finally used to map the gray level of the pixel centered in the neighborhood. The center of the neighborhood region is then moved to an adjacent pixel location and the procedure is repeated. Since only one new row or column of the neighborhood changes during a pixel-to-pixel translation of the region, updating the histogram obtained in the previous location with the new data introduced at each motion step is possible (Problem 3.11). This approach has obvious advantages over repeatedly computing the histogram over all pixels in the neighborhood region each time the region is moved one pixel location. Another approach used some times to reduce computation is to utilize nonoverlapping regions, but this method usually produces an undesirable checkerboard effect.

EXAMPLE 3.5: Enhancement using local histograms.

Figure 3.23(a) shows an image that has been slightly blurred to reduce its noise content (see Section 3.6.1 regarding blurring). Figure 3.23(b) shows the result of global histogram equalization. As is often the case when this technique is applied to smooth, noisy areas, Fig. 3.23(b) shows considerable enhancement of the noise, with a slight increase in contrast. Note that no new structural details were brought out by this method. However, local histogram equalization using a 7×7 neighborhood revealed the presence of small squares inside the larger dark squares. The small squares were too close in gray level to the larger ones, and their sizes were too small to influence global histogram equalization significantly. Note also the finer noise texture in Fig. 3.23(c), a result of local processing using relatively small neighborhoods.

Use of Histogram Statistics for Image Enhancement

Instead of using the image histogram directly for enhancement, we can use instead some statistical parameters obtainable directly from the histogram. Let r denote a discrete random variable representing discrete gray-levels in the range



a b c

FIGURE 3.23 (a) Original image. (b) Result of global histogram equalization. (c) Result of local histogram equalization using a 7 × 7 neighborhood about each pixel.

$[0, L-1]$, and let $p(r_i)$ denote the normalized histogram component corresponding to the ith value of r. As indicated previously in this section, we may view $p(r_i)$ as an estimate of the probability of occurrence of gray level r_i. The nth moment of r about its mean is defined as

$$\mu_n(r) = \sum_{i=0}^{L-1} (r_i - m)^n p(r_i) \tag{3.3-18}$$

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

$$m = \sum_{i=0}^{L-1} r_i p(r_i). \tag{3.3-19}$$

It follows from Eqs. (3.3-18) and (3.3-19) that $\mu_0 = 1$ and $\mu_1 = 0$. The second moment is given by

$$\mu_2(r) = \sum_{i=0}^{L-1} (r_i - m)^2 p(r_i). \tag{3.3-20}$$

We recognize this expression as the variance of r, which is denoted conventionally by $\sigma^2(r)$. The standard deviation is defined simply as the square root of the variance. We will revisit moments in Chapter 11 in connection with image description. In terms of enhancement, however, we are interested primarily in the mean, which is a measure of average gray level in an image, and the variance (or standard deviation), which is a measure of average contrast.

We consider two uses of the mean and variance for enhancement purposes. The *global* mean and variance are measured over an entire image and are useful primarily for gross adjustments of overall intensity and contrast. A much more powerful use of these two measures is in local enhancement, where the *local* mean and variance are used as the basis for making changes that depend on image characteristics in a predefined region about each pixel in the image.

Let (x, y) be the coordinates of a pixel in an image, and let S_{xy} denote a neighborhood (subimage) of specified size, centered at (x, y). From Eq. (3.3-19) the mean value $m_{S_{xy}}$ of the pixels in S_{xy} can be computed using the expression

$$m_{S_{xy}} = \sum_{(s,t)\in S_{xy}} r_{s,t} p(r_{s,t}) \tag{3.3-21}$$

where $r_{s,t}$ is the gray level at coordinates (s, t) in the neighborhood, and $p(r_{s,t})$ is the neighborhood normalized histogram component corresponding to that value of gray level. Similarly, from Eq. (3.3-20), the gray-level variance of the pixels in region S_{xy} is given by

$$\sigma^2_{S_{xy}} = \sum_{(s,t)\in S_{xy}} [r_{s,t} - m_{S_{xy}}]^2 p(r_{s,t}). \tag{3.3-22}$$

The local mean is a measure of average gray level in neighborhood S_{xy}, and the variance (or standard deviation) is a measure of contrast in that neighborhood.

An important aspect of image processing using the local mean and variance is the flexibility they afford in developing simple, yet powerful enhancement techniques based on statistical measures that have a close, predictable correspondence with image appearance. We illustrate these characteristics by means of an example.

EXAMPLE 3.6: Enhancement based on local statistics.

Figure 3.24 shows an SEM (scanning electron microscope) image of a tungsten filament wrapped around a support. The filament in the center of the image and its support are quite clear and easy to study. There is another filament structure on the right side of the image, but it is much darker and its size and other features are not as easily discernable. Local enhancement by contrast manipulation is an ideal approach to try on problems such as this, where part of the image is acceptable, but other parts may contain hidden features of interest.

In this particular case, the problem is to enhance dark areas while leaving the light area as unchanged as possible since it does note require enhancement. We can use the concepts presented in this section to formulate an enhancement method that can tell the difference between dark and light and, at the same time, is capable of enhancing only the dark areas. A measure of whether an area is relatively light or dark at a point (x, y) is to compare the local average gray level $m_{S_{xy}}$ to the average image gray level, called the global mean and denoted M_G. This latter quantity is obtained by letting S encompass the entire image. Thus, we have the first element of our enhancement scheme: We will consider the pixel at a point (x, y) as a candidate for processing if $m_{S_{xy}} \leq k_0 M_G$, where k_0 is a positive constant with value less than 1.0. Since we are interested in enhancing areas that have low contrast, we also need a measure to determine whether the contrast of an area makes it a candidate for enhancement. Thus, we will consider the pixel at a point (x, y) as a candidate for enhancement if $\sigma_{S_{xy}} \leq k_2 D_G$, where D_G is the global standard deviation and k_2 is a positive constant. The value of this constant will be greater than 1.0 if we are interested in enhancing light areas and less than 1.0 for dark areas. Finally, we need to restrict

the lowest values of contrast we are willing to accept, otherwise the procedure would attempt to enhance even constant areas, whose standard deviation is zero. Thus, we also set a lower limit on the local standard deviation by requiring that $k_1 D_G \leq \sigma_{S_{xy}}$, with $k_1 < k_2$. A pixel at (x, y) that meets all the conditions for local enhancement is processed simply by multiplying it by a specified constant, E, to increase (or decrease) the value of its gray level relative to the rest of the image. The values of pixels that do not meet the enhancement conditions are left unchanged.

A summary of the enhancement method is as follows. Let $f(x, y)$ represent the value of an image pixel at any image coordinates (x, y), and let $g(x, y)$ represent the corresponding enhanced pixel at those coordinates. Then

$$g(x, y) = \begin{cases} E \cdot f(x, y) & \text{if } m_{S_{xy}} \leq k_0 M_G \quad \text{AND} \quad k_1 D_G \leq \sigma_{S_{xy}} \leq k_2 D_G \\ f(x, y) & \text{otherwise} \end{cases}$$

where, as indicated previously, E, k_0, k_1, and k_2 are specified parameters; M_G is the global mean of the input image; and D_G is its global standard deviation.

Normally, making a successful selection of parameters requires a bit of experimentation to gain familiarity with a given image or class of images. In this case, the following values were selected: $E = 4.0$, $k_0 = 0.4$, $k_1 = 0.02$, and $k_2 = 0.4$. The relatively low value of 4.0 for E was chosen so that, when it was multiplied by the levels in the areas being enhanced (which are dark), the result would still tend toward the dark end of the scale, and thus preserve the general visual balance of the image. The value of k_0 was chosen as somewhat less than half the global mean since it is obvious by looking at the image that the areas that require enhancement definitely are dark enough to be below half the global mean. A similar analysis led to the choice of values for k_1 and k_2. Choosing these constants is not a difficult task in general, but their choice

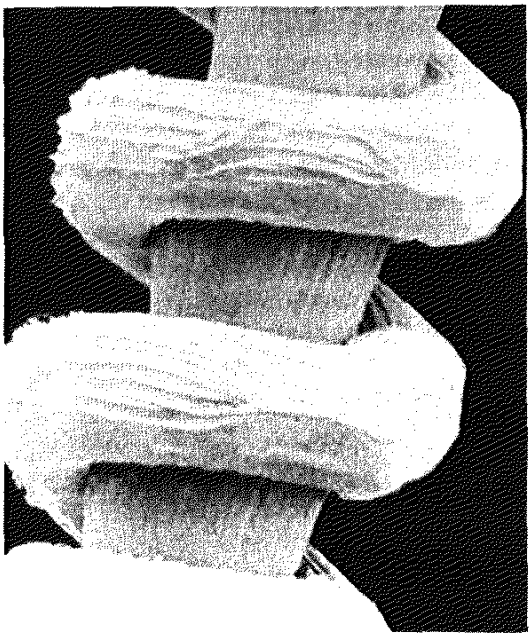
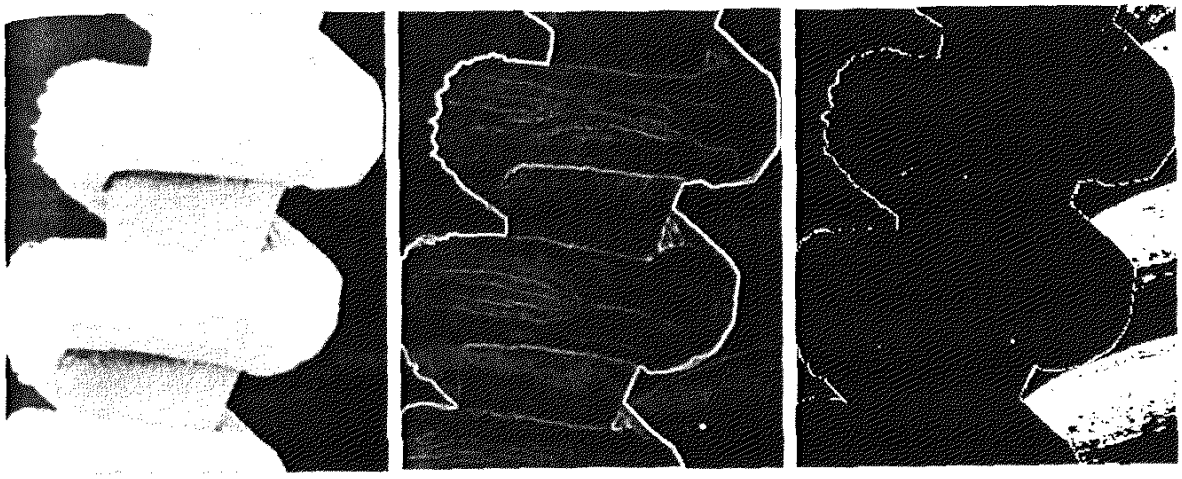


FIGURE 3.24 SEM image of a tungsten filament and support, magnified approximately 130×. (Original image courtesy of Mr. Michael Shaffer, Department of Geological Sciences, University of Oregon, Eugene).



a b c

FIGURE 3.25 (a) Image formed from all local means obtained from Fig. 3.24 using Eq. (3.3-21). (b) Image formed from all local standard deviations obtained from Fig. 3.24 using Eq. (3.3-22). (c) Image formed from all multiplication constants used to produce the enhanced image shown in Fig. 3.26.

definitely must be guided by a logical analysis of the enhancement problem at hand. Finally, the choice of size for the local area should be as small as possible in order to preserve detail and keep the computational burden as low as possible. We chose a small (3×3) local region.

Figure 3.25(a) shows the values of $m_{S_{xy}}$ for all values of (x, y). Since the value of $m_{S_{xy}}$ for each (x, y) is the average of the neighboring pixels in a 3×3 area centered at (x, y), we expect the result to be similar to the original image, but

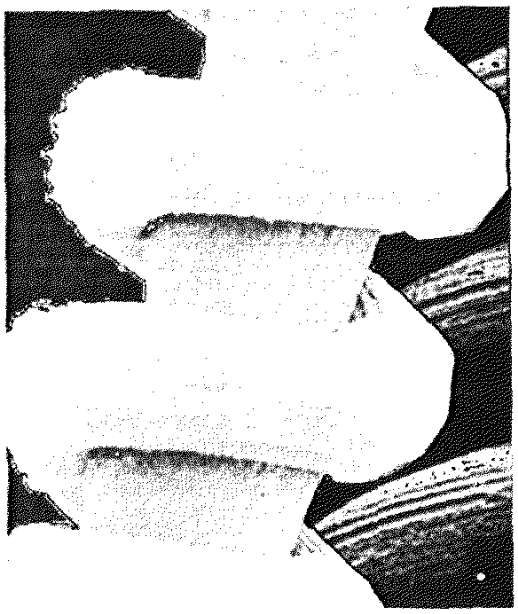


FIGURE 3.26 Enhanced SEM image. Compare with Fig. 3.24. Note in particular the enhanced area on the right side of the image.

slightly blurred. This indeed is the case in Fig. 3.25(a). Figure 3.25(b) shows in image formed using all the values of $\sigma_{S_{xy}}$. Similarly, we can construct an image out the values that multiply $f(x, y)$ at each coordinate pair (x, y) to form $g(x, y)$. Since the values are either 1 or E, the image is binary, as shown in Fig. 3.25(c). The dark areas correspond to 1 and the light areas to E. Thus, any light point in Fig. 3.25(c) signifies a coordinate pair (x, y) at which the enhancement procedure multiplied $f(x, y)$ by E to produce an enhanced pixel. The dark points represent coordinates at which the procedure did not to modify the pixel values.

The enhanced image obtained with the method just described is shown in Fig. 3.26. In comparing this image with the original in Fig. 3.24, we note the obvious detail that has been brought out on the right side of the enhanced image. It is worthwhile to point out that the unenhanced portions of the image (the light areas) were left intact for the most part. We do note the appearance of some small bright dots in the shadow areas where the coil meets the support stem, and around some of the borders between the filament and the background. These are undesirable artifacts created by the enhancement technique. In other words, the points appearing as light dots met the criteria for enhancement and their values were amplified by factor E. Introduction of artifacts is a definite drawback of a method such as the one just described because of the nonlinear way in which they process an image. The key point here, however, is that the image was enhanced in a most satisfactory way as far as bringing out the desired detail. ■

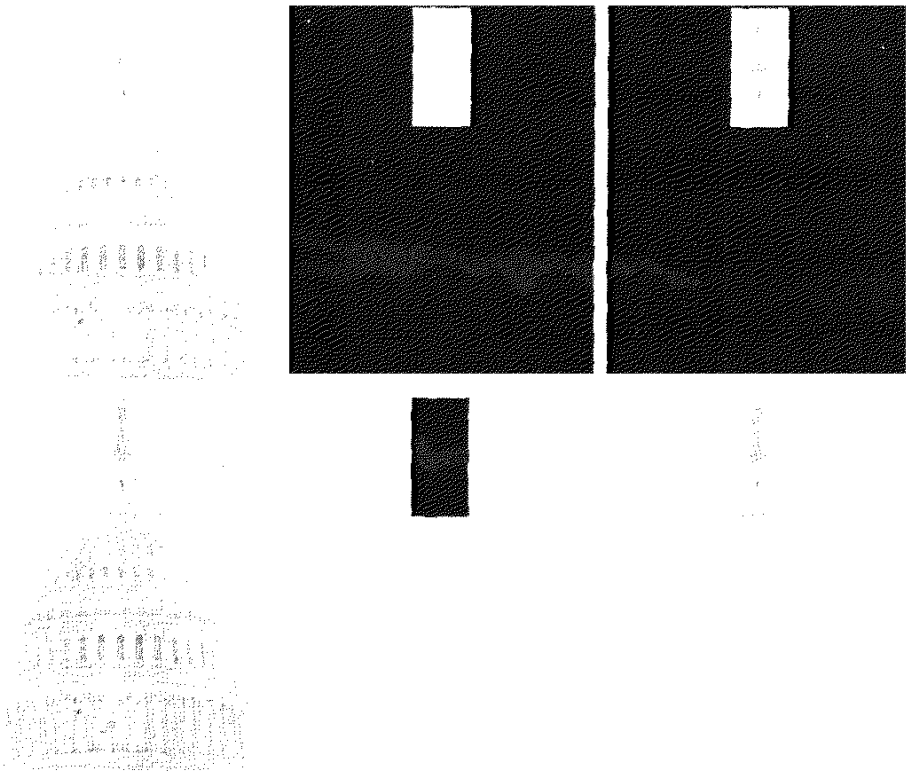
It is not difficult to imagine the numerous ways in which the example just given could be adapted or extended to other situations in which local enhancement is applicable.

3.4 Enhancement Using Arithmetic/Logic Operations

Arithmetic/logic operations involving images are performed on a pixel-by-pixel basis between two or more images (this excludes the logic operation NOT, which is performed on a single image). As an example, subtraction of two images results in a new image whose pixel at coordinates (x, y) is the difference between the pixels in that same location in the two images being subtracted. Depending on the hardware and/or software being used, the actual mechanics of implementing arithmetic/logic operations can be done sequentially, one pixel at a time, or in parallel, where all operations are performed simultaneously.

Logic operations similarly operate on a pixel-by-pixel basis†. We need only be concerned with the ability to implement the AND, OR, and NOT logic operators because these three operators are *functionally complete*. In other words, any other logic operator can be implemented by using only these three basic functions. When dealing with logic operations on gray-scale images, pixel values are processed as strings of binary numbers. For example, performing the NOT operation on a black, 8-bit pixel (a string of eight 0's) produces a white pixel

† Recall that, for two binary variables a and b: aANDb yields 1 only when both a and b are 1; otherwise the result is 0. Similarly, aORb is 0 when both variables are 0; otherwise the result is 1. Finally, if a is 1, NOT (a) is 0, and vice versa.



a b c
d e f

FIGURE 3.27 (a) Original image. (b) AND image mask. (c) Result of the AND operation on images (a) and (b). (d) Original image. (e) OR image mask. (f) Result of operation OR on images (d) and (e).

(a string of eight 1's). Intermediate values are processed the same way, changing all 1's to 0's and vice versa. Thus, the NOT logic operator performs the same function as the negative transformation of Eq. (3.2-1). The AND and OR operations are used for *masking*; that is, for selecting subimages in an image, as illustrated in Fig. 3.27. In the AND and OR image masks, light represents a binary 1 and dark represents a binary 0. Masking sometimes is referred to as *region of interest* (ROI) processing. In terms of enhancement, masking is used primarily to isolate an area for processing. This is done to highlight that area and differentiate it from the rest of the image. Logic operations also are used frequently in conjunction with morphological operations, as discussed in Chapter 9.

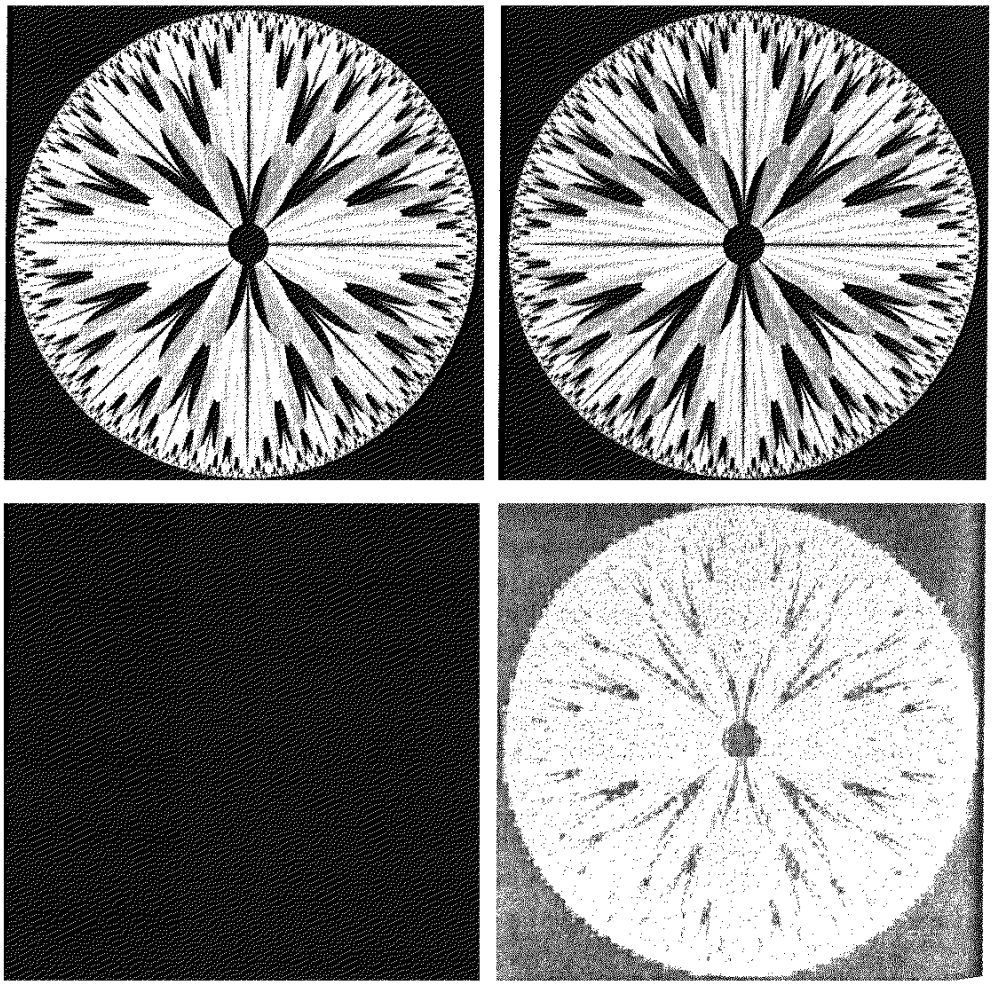
Of the four arithmetic operations, subtraction and addition (in that order) are the most useful for image enhancement. We consider division of two images simply as multiplication of one image by the reciprocal of the other. Aside from the obvious operation of multiplying an image by a constant to increase its average gray level, image multiplication finds use in enhancement primarily as a masking operation that is more general than the logical masks discussed in the previous paragraph. In other words, multiplication of one image by another can be used to implement gray-level, rather than binary, masks. We give an example in Section 3.8 of how such a masking operation can be a useful tool. In the remainder of this section, we develop and illustrate methods based on subtraction and addition for image enhancement. Other uses of image multiplication are discussed in Chapter 5, in the context of image restoration.

3.4.1 Image Subtraction

The difference between two images $f(x, y)$ and $h(x, y)$, expressed as

$$g(x, y) = f(x, y) - h(x, y), \tag{3.4-1}$$

is obtained by computing the difference between all pairs of corresponding pixels from f and h. The key usefulness of subtraction is the enhancement of *differences* between images. We illustrate this concept by returning briefly to the discussion in Section 3.2.4, where we showed that the higher-order bit planes of an image carry a significant amount of visually relevant detail, while the lower planes contribute more to fine (often imperceptible) detail. Figure 3.28(a) shows the fractal image used earlier to illustrate the concept of bit planes. Figure 3.28(b) shows the result of discarding (setting to zero) the four least significant bit planes of the original image. The images are nearly identical visually, with the exception of a very slight drop in overall contrast due to less variability of the gray-level values in the image of Fig. 3.28(b). The pixel-by-pixel difference between these two images is shown in Fig. 3.28(c). The differences in pixel values are so small that the difference image appears nearly black when displayed on an 8-bit



a b
c d

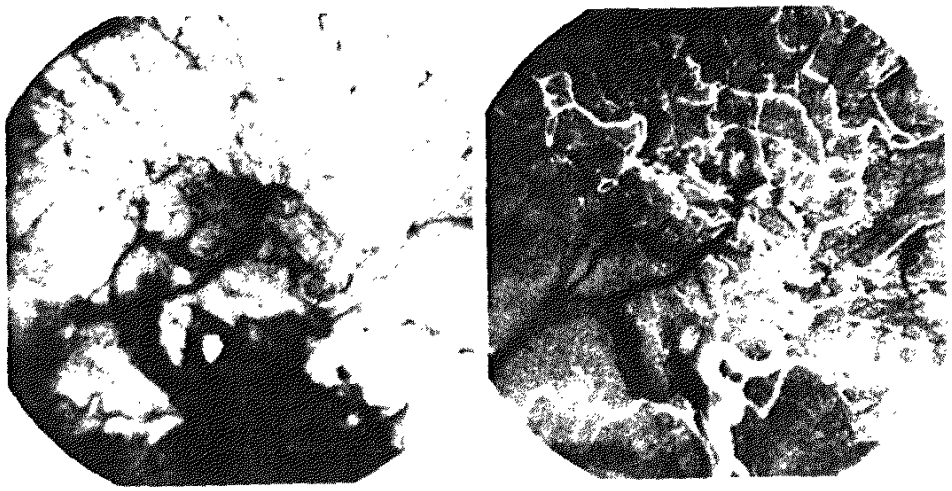
FIGURE 3.28
(a) Original fractal image. (b) Result of setting the four lower-order bit planes to zero. (c) Difference between (a) and (b). (d) Histogram-equalized difference image. (Original image courtesy of Ms. Melissa D. Binde, Swarthmore College, Swarthmore, PA).

display. In order to bring out more detail, we can perform a contrast stretching transformation, such as those discussed in Sections 3.2 or 3.3. We chose histogram equalization, but an appropriate power-law transformation would have done the job also. The result is shown in Fig. 3.28(d). This is a very useful image for evaluating the effect of setting to zero the lower-order planes.

EXAMPLE 3.7: Use of image subtraction in mask mode radiography.

One of the most commercially successful and beneficial uses of image subtraction is in the area of medical imaging called *mask mode radiography*. In this case $h(x, y)$, the *mask*, is an X-ray image of a region of a patient's body captured by an intensified TV camera (instead of traditional X-ray film) located opposite an X-ray source. The procedure consists of injecting a contrast medium into the patient's bloodstream, taking a series of images of the same anatomical region as $h(x, y)$, and subtracting this mask from the series of incoming images after injection of the contrast medium. The net effect of subtracting the mask from each sample in the incoming stream of TV images is that the areas that are different between $f(x, y)$ and $h(x, y)$ appear in the output image as enhanced detail. Because images can be captured at TV rates, this procedure in essence gives a movie showing how the contrast medium propagates through the various arteries in the area being observed.

Figure 3.29(a) shows an X-ray image of the top of a patient's head prior to injection of an iodine medium into the bloodstream. The camera yielding this image was positioned above the patient's head, looking down. As a reference point, the bright spot in the lower one-third of the image is the core of the spinal column. Figure 3.29(b) shows the difference between the mask (Fig. 3.29a) and an image taken some time after the medium was introduced into the bloodstream. The bright arterial paths carrying the medium are unmistakably enhanced in Fig. 3.29(b). These arteries appear quite bright because they are not subtracted out (that is, they are not part of the mask image). The overall background is much darker than that in Fig. 3.29(a) because differences between areas of little change yield low values, which in turn appear as dark shades of gray in the difference image. Note, for instance, that the spinal cord, which is bright in Fig. 3.29(a), appears quite dark in Fig. 3.29(b) as a result of subtraction.



a b

FIGURE 3.29 Enhancement by image subtraction. (a) Mask image. (b) An image (taken after injection of a contrast medium into the bloodstream) with mask subtracted out.

A few comments on implementation are an order before we leave this section. In practice, most images are displayed using 8 bits (even 24-bit color images consists of three separate 8-bit channels). Thus, we expect image values not to be outside the range from 0 to 255. The values in a difference image can range from a minimum of −255 to a maximum of 255, so some sort of scaling is required to display the results. There are two principal ways to scale a difference image. One method is to add 255 to every pixel and then divide by 2. It is not guaranteed that the values will cover the entire 8-bit range from 0 to 255, but all pixel values definitely will be within this range. This method is fast and simple to implement, but it has the limitations that the full range of the display may not be utilized and, potentially more serious, the truncation inherent in the division by 2 will generally cause loss in accuracy.

If more accuracy and full coverage of the 8-bit range are desired, then we can resort to another approach. First, the value of the minimum difference is obtained and its negative added to all the pixels in the difference image (this will create a modified difference image whose minimum values is 0). Then, all the pixels in the image are scaled to the interval [0, 255] by multiplying each pixel by the quantity 255/Max, where Max is the maximum pixel value in the modified difference image. It is evident that this approach is considerably more complex and difficult to implement.

Before leaving this section we note also that change detection via image subtraction finds another major application in the area of segmentation, which is the topic of Chapter 10. Basically, segmentation techniques attempt to subdivide an image into regions based on a specified criterion. Image subtraction for segmentation is used when the criterion is "changes." For instance, in tracking (segmenting) moving vehicles in a sequence of images, subtraction is used to remove all stationary components in an image. What is left should be the moving elements in the image, plus noise.

3.4.2 Image Averaging

Consider a noisy image $g(x, y)$ formed by the addition of noise $\eta(x, y)$ to an original image $f(x, y)$; that is,

$$g(x, y) = f(x, y) + \eta(x, y) \tag{3.4-2}$$

where the assumption is that at every pair of coordinates (x, y) the noise is uncorrelated[†] and has zero average value. The objective of the following procedure is to reduce the noise content by adding a set of noisy images, $\{g_i(x, y)\}$.

If the noise satisfies the constraints just stated, it can be shown (Problem 3.15) that if an image $\bar{g}(x, y)$ is formed by averaging K different noisy images,

$$\bar{g}(x, y) = \frac{1}{K} \sum_{i=1}^{K} g_i(x, y) \tag{3.4-3}$$

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

then it follows that

$$E\{\bar{g}(x, y)\} = f(x, y) \tag{3.4-4}$$

and

$$\sigma^2_{\bar{g}(x,y)} = \frac{1}{K}\sigma^2_{\eta(x,y)} \tag{3.4-5}$$

where $E\{\bar{g}(x, y)\}$ is the expected value of $\bar{g}$, and $\sigma^2_{\bar{g}(x,y)}$ and $\sigma^2_{\eta(x,y)}$ are the variances of $\bar{g}$ and η, all at coordinates (x, y). The standard deviation at any point in the average image is

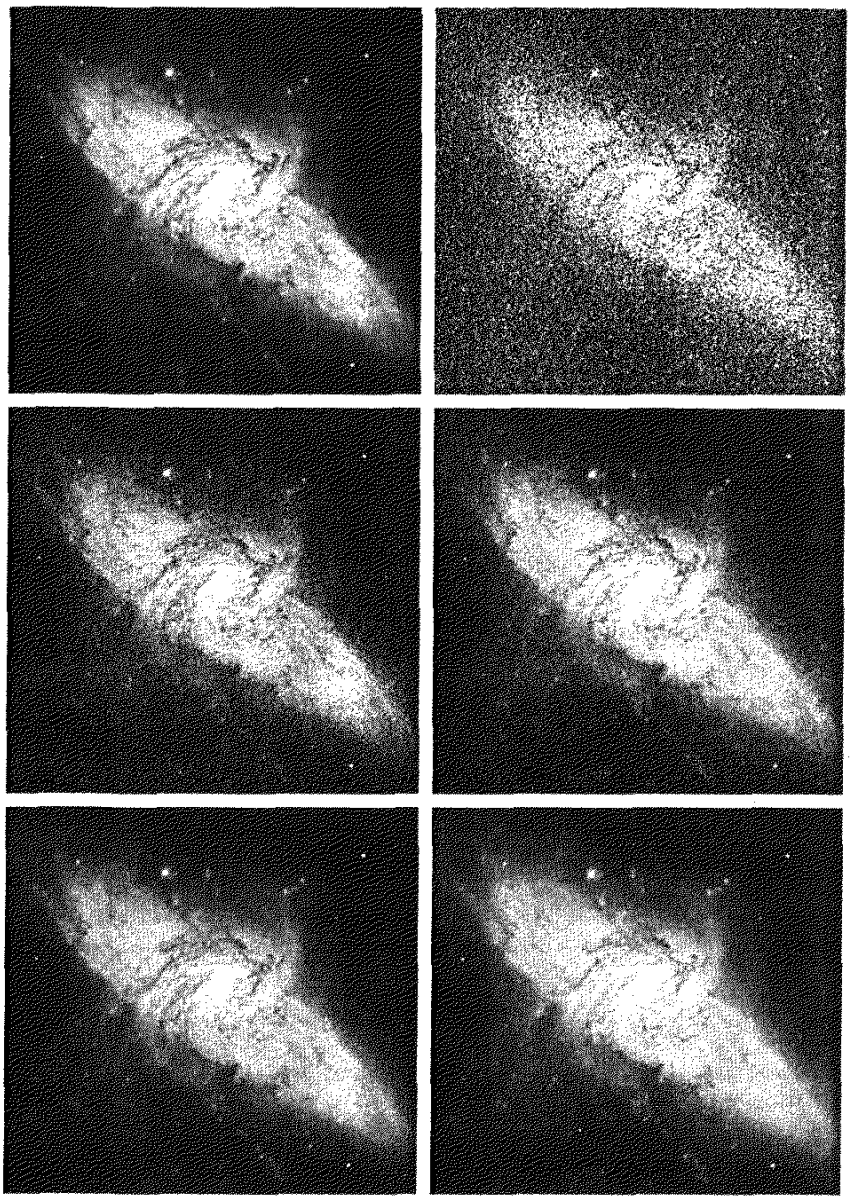
$$\sigma_{\bar{g}(x,y)} = \frac{1}{\sqrt{K}}\sigma_{\eta(x,y)}. \tag{3.4-6}$$

As K increases, Eqs. (3.4-5) and (3.4-6) indicate that the variability (noise) of the pixel values at each location (x, y) decreases. Because $E\{\bar{g}(x, y)\} = f(x, y)$, this means that $\bar{g}(x, y)$ approaches $f(x, y)$ as the number of noisy images used in the averaging process increases. In practice, the images $g_i(x, y)$ must be registered (aligned) in order to avoid the introduction of blurring and other artifacts in the output image.

EXAMPLE 3.8: Noise reduction by image averaging.

■ An important application of image averaging is in the field of astronomy, where imaging with very low light levels is routine, causing sensor noise frequently to render single images virtually useless for analysis. Figure 3.30(a) shows an image of a galaxy pair called NGC 3314, taken by NASA's Hubble Space Telescope with a wide field planetary camera. NGC 3314 lies about 140 million light-years from Earth, in the direction of the southern-hemisphere constellation Hydra. The bright stars forming a pinwheel shape near the center of the front galaxy have formed recently from interstellar gas and dust. Figure 3.30(b) shows the same image, but corrupted by uncorrelated Gaussian noise with zero mean and a standard deviation of 64 gray levels. This image is useless for all practical purposes. Figures 3.30(c) through (f) show the results of averaging 8, 16, 64, and 128 images, respectively. We see that the result obtained with $K = 128$ is reasonably close to the original in visual appearance.

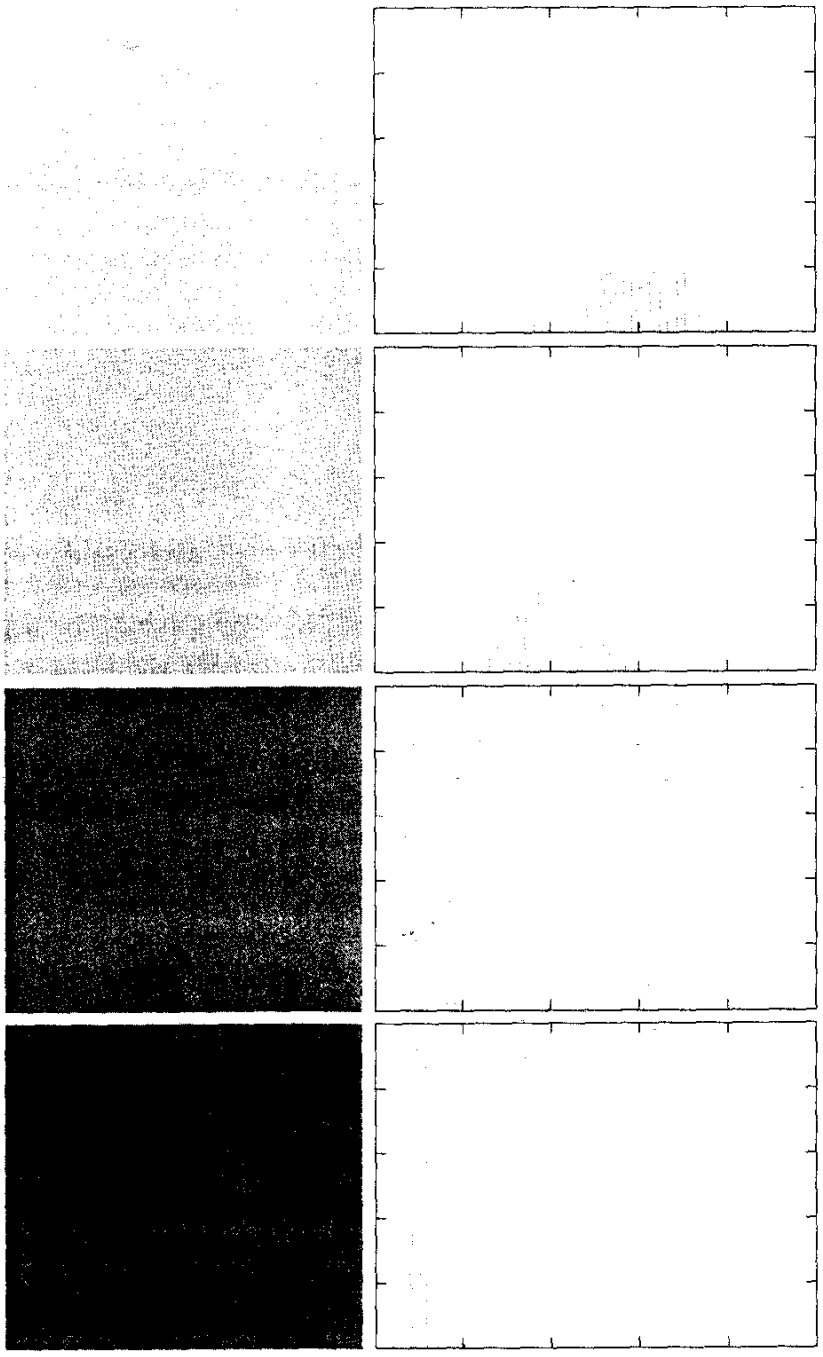
We can get a better appreciation from Fig. 3.31 for how reduction in the visual appearance of noise takes place as a function of increasing K. This figure shows the difference images between the original [Fig. 3.30(a)] and each of the averaged images in Figs. 3.30(c) through (f). The histograms corresponding to the difference images are also shown in the figure. As usual, the vertical scale in the histograms represents number of pixels and is in the range $[0, 2.6 \times 10^4]$. The horizontal scale represents gray level and is in the range $[0, 255]$. Notice in the histograms that the mean and standard deviation of the difference images decrease as K increases. This is as expected because, according to Eqs. (3.4-3) and (3.4-4), the average image should approach the original as K increases. We can also see the effect of a decreasing mean in the difference images on the left column of Fig. 3.31, which become darker as the K increases.



a b
c d
e f

FIGURE 3.30 (a) Image of Galaxy Pair NGC 3314. (b) Image corrupted by additive Gaussian noise with zero mean and a standard deviation of 64 gray levels. (c)–(f) Results of averaging $K = 8, 16, 64$, and 128 noisy images. (Original image courtesy of NASA.)

Addition is the discrete formulation of continuous integration. In astronomical observations, a process equivalent to the method just described is to use the integrating capabilities of CCD or similar sensors for noise reduction by observing the same scene over long periods of time. The net effect, however, is analogous to the procedure just discussed. Cooling the sensor further reduces its noise level.



a b

FIGURE 3.31
(a) From top to bottom: Difference images between Fig. 3.30(a) and the four images in Figs. 3.30(c) through (f), respectively. (b) Corresponding histograms.

As in the case of image subtraction, adding two or more 8-bit images requires special care when it comes to displaying the result on an 8-bit display. The values in the sum of K, 8-bit images can range from 0 to $255 \times K$. Scaling back to 8 bits in this case consists simply of dividing the result by K. Naturally, some accuracy will be lost in the process, but this is unavoidable if the display has to be limited to 8 bits.

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/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 3×3 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 3×3 (we exclude from our discussion the trivial case of a 1×1 mask).

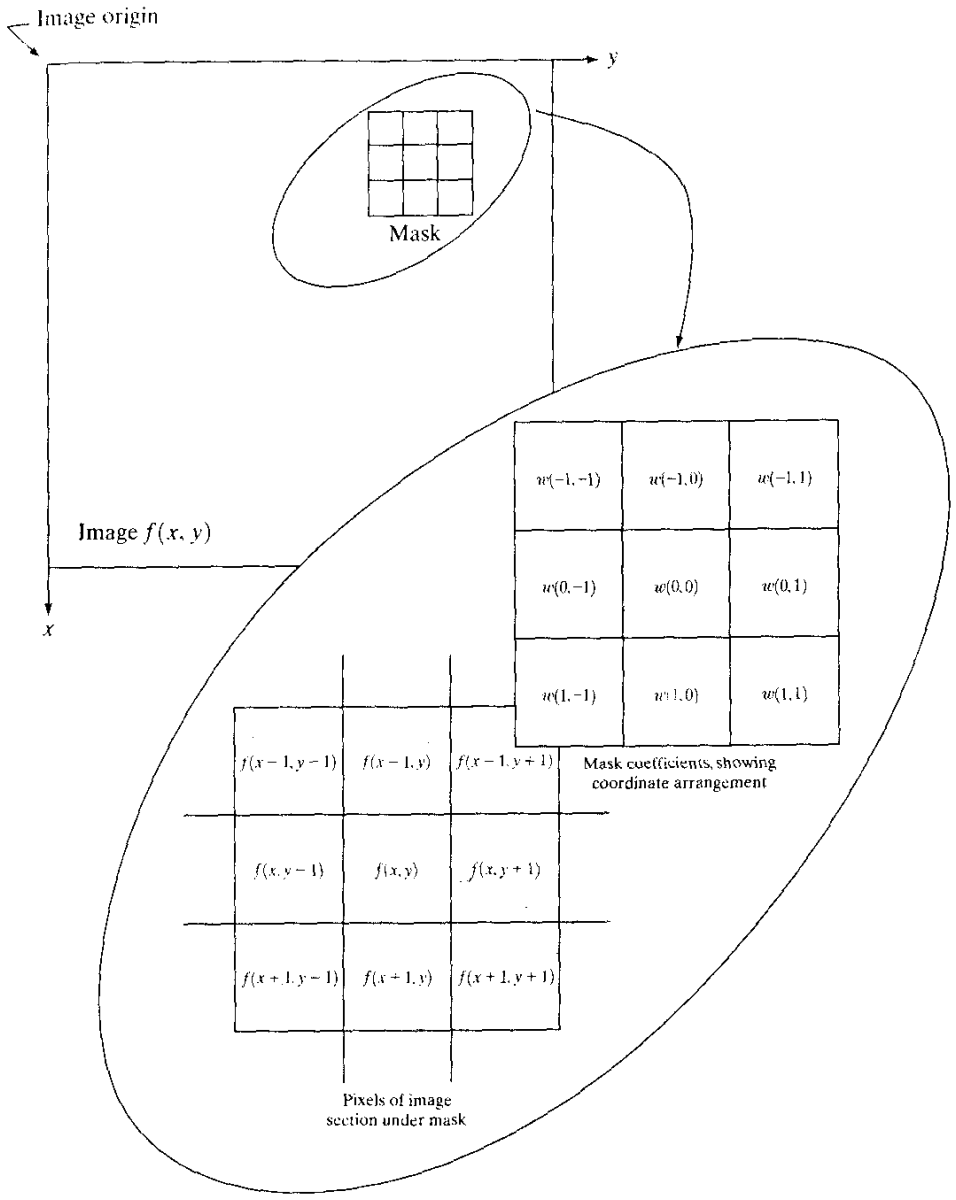


FIGURE 3.32 The mechanics of spatial filtering. The magnified drawing shows a 3×3 mask and the image section directly under it; the image section is shown displaced out from under the mask for ease of readability.

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:

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

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 $x = 0, 1, 2, \ldots, M - 1$ and $y = 0, 1, 2, \ldots, N - 1$. 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:

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

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 3×3 general mask shown in Fig. 3.33 the response at any point (x, y) in the image is given by

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

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.

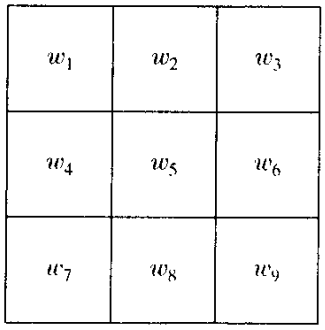


FIGURE 3.33 Another representation of a general 3×3 spatial filter mask.

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 imaged 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 nonlinear 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 a *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 3×3 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.

$$\frac{1}{9} \times \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \qquad \frac{1}{16} \times \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

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 3×3 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 3×3 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

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

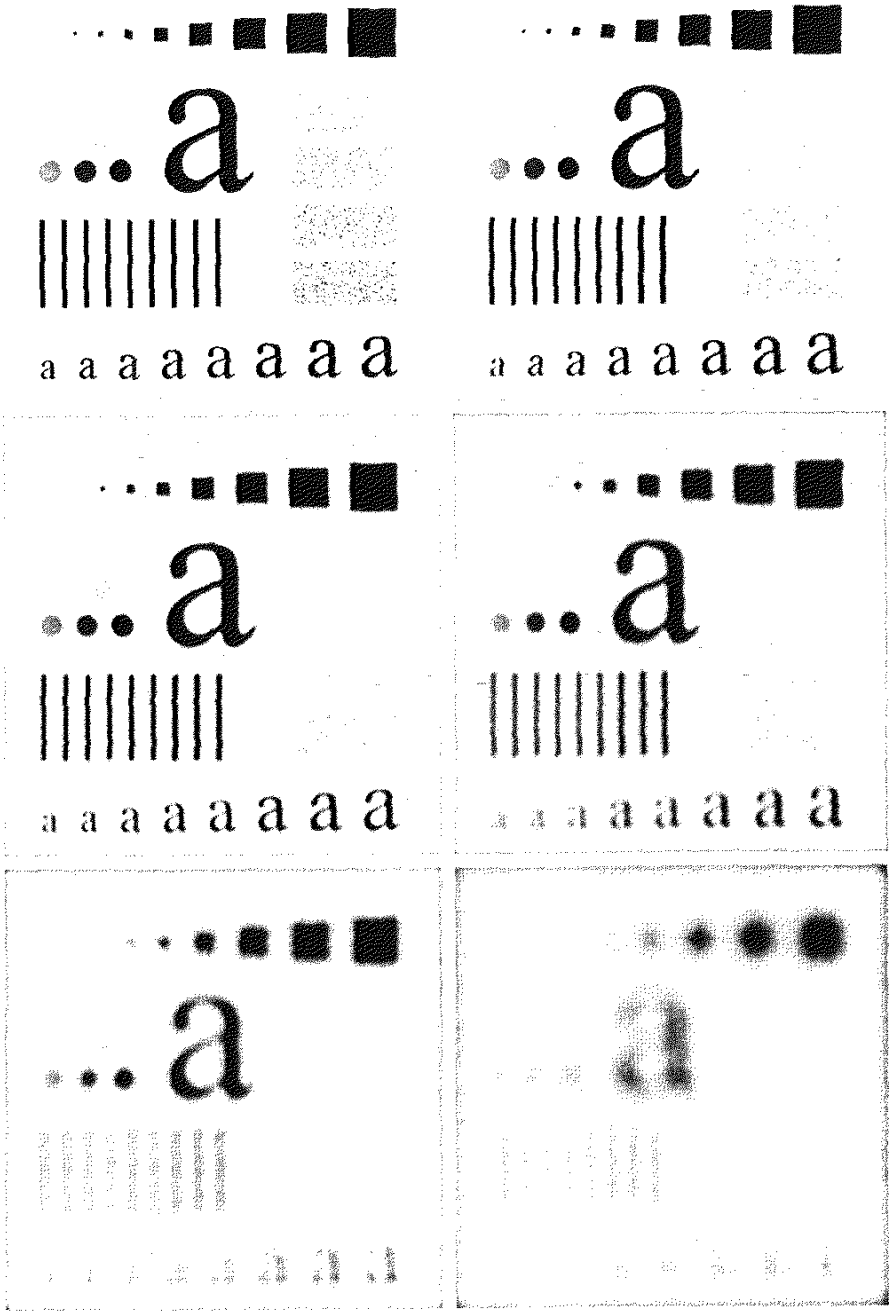
The parameters in this equation are as defined in Eq. (3.5-1). As before, it is understood that the complete filtered image is obtained by applying Eq. (3.6-1) for $x = 0, 1, 2, \ldots, M - 1$ and $y = 0, 1, 2, \ldots, N - 1$. The denominator in Eq. (3.6-1) is simply the sum of the mask coefficients 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.

EXAMPLE 3.9: Image smoothing with masks of various sizes.

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 3×3 and 5×5 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.

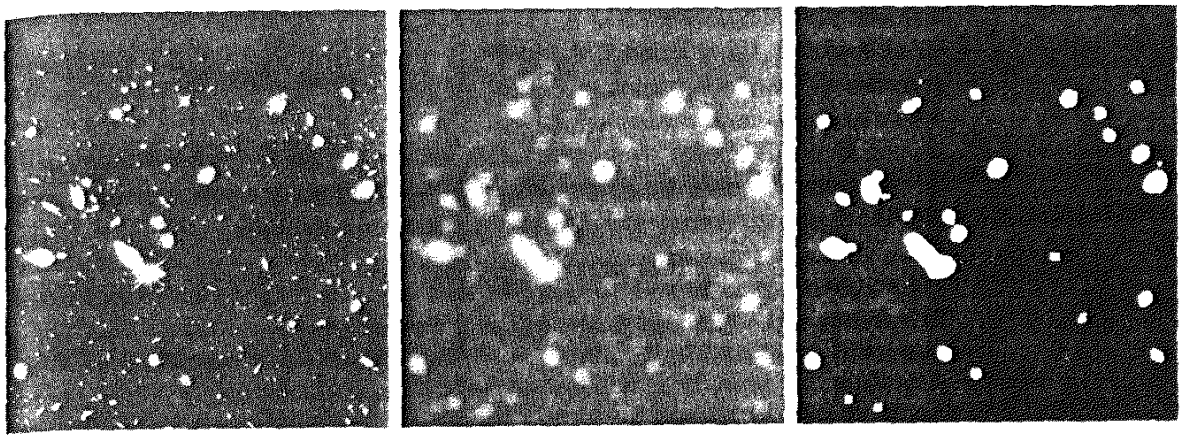
The result for $n = 5$ is somewhat similar, with a slight further increase in blurring. For $n = 9$ we see considerably more blurring, and the 20% black circle is not nearly as distinct from the background as in the previous three images, illustrating the blending effect that blurring has on objects 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. ■

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



a b
c d
e f

FIGURE 3.35 (a) Original image, of size 500 × 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 × 120 pixels.



a b c

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

15 × 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.

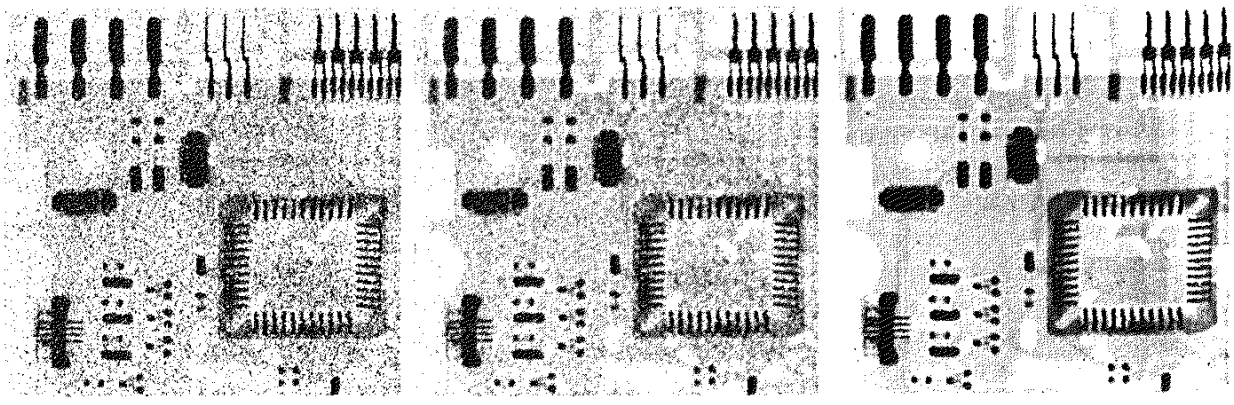
The median, ξ, of a set of values is such that half the values in the set are less than or equal to ξ, and half are greater than or equal to ξ. In order to perform median filtering at a point in an image, we first sort the values of the pixel in question and its neighbors, determine their median, and assign this value to that pixel. For example, in a 3 × 3 neighborhood the median is the 5th largest value, in a 5 × 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 3×3 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 3×3 max filter is given by $R = \max\{z_k | k = 1, 2, \ldots, 9\}$. 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 3×3 neighborhood averaging mask, and in Fig. 3.37(c) the result of using a 3×3 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.



a b c

FIGURE 3.37 (a) X-ray image of circuit board corrupted by salt-and-pepper noise. (b) Noise reduction with a 3×3 averaging mask. (c) Noise reduction with a 3×3 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

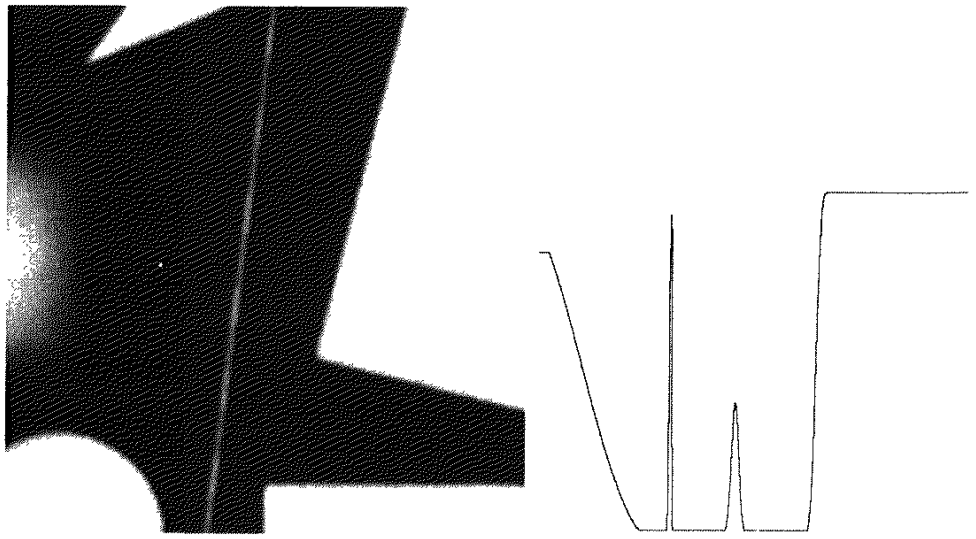
$$\frac{\partial^2 f}{\partial x^2} = f(x + 1) + f(x - 1) - 2f(x).$$

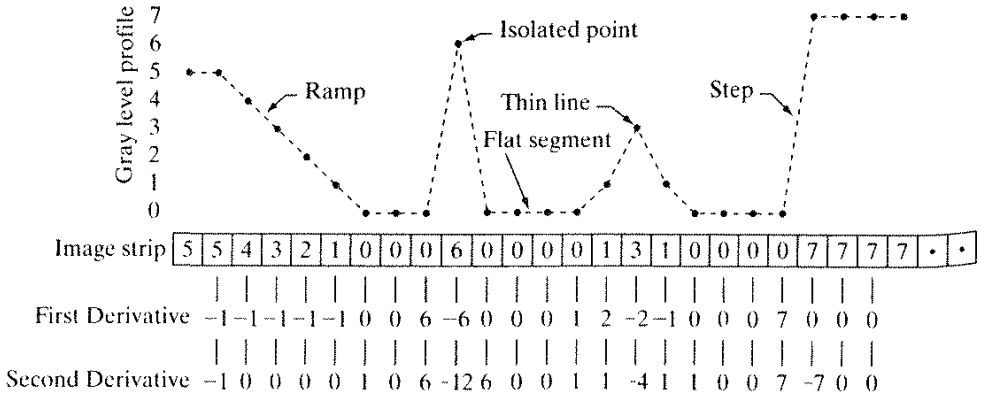
It is easily verified that these two definitions satisfy the conditions stated previously regarding derivatives of the first and second order. To see this, and also to highlight the fundamental similarities and differences between first- and second-order derivatives in the context of image processing,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

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

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:

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

and, similarly in the y-direction, as

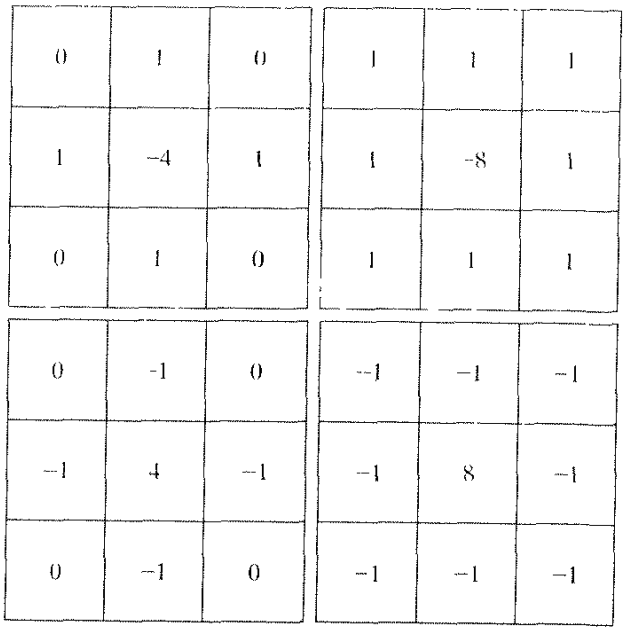
$$\frac{\partial^2 f}{\partial^2 y^2} = f(x, y + 1) + f(x, y - 1) - 2f(x, y) \tag{3.7-3}$$

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

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

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°. 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°. 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:

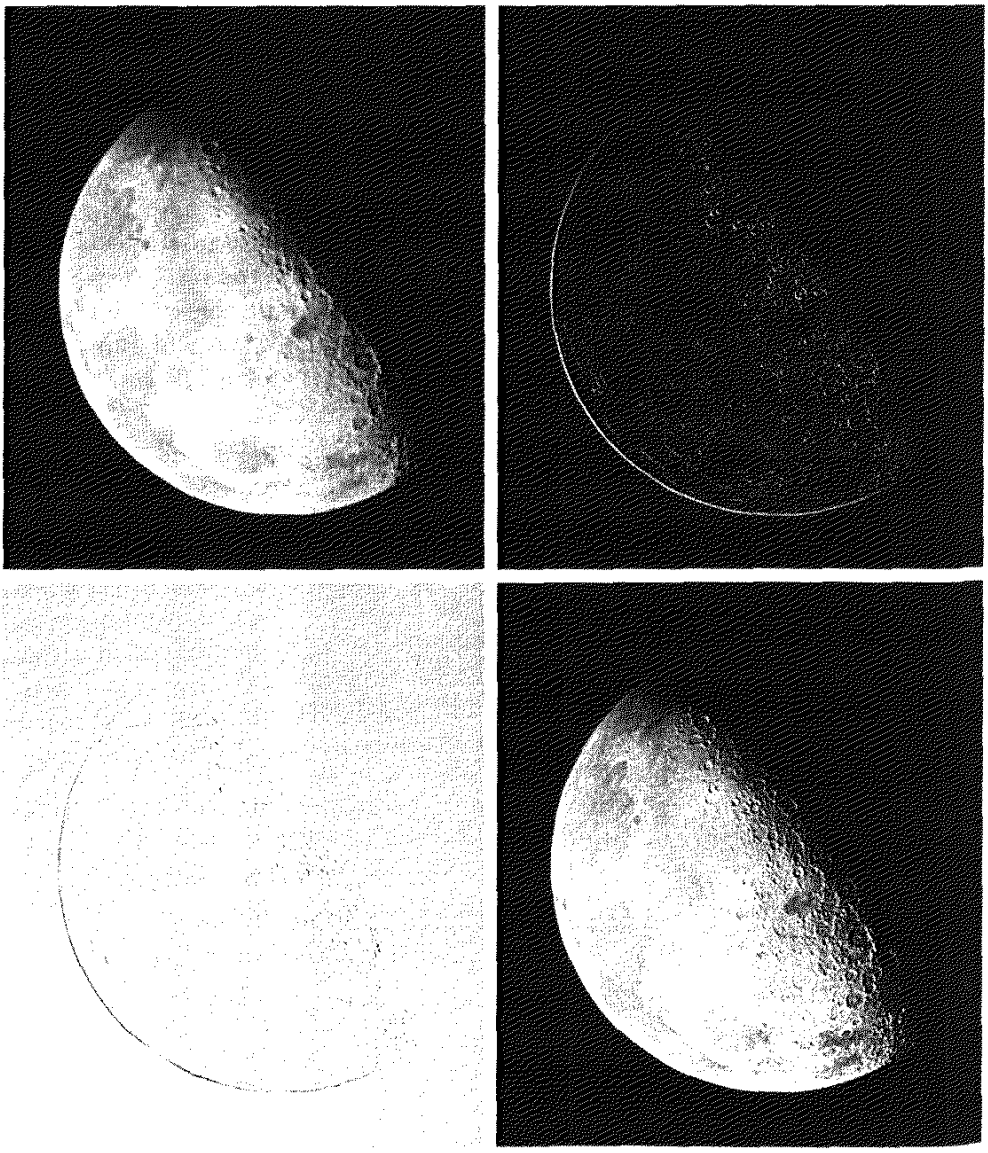
$$g(x, y) = \begin{cases} f(x, y) - \nabla^2 f(x, y) & \text{if the center coefficient of the Laplacian mask is negative} \\ f(x, y) + \nabla^2 f(x, y) & \text{if the center coefficient of the Laplacian mask is positive.} \end{cases} \tag{3.7-5}$$

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):

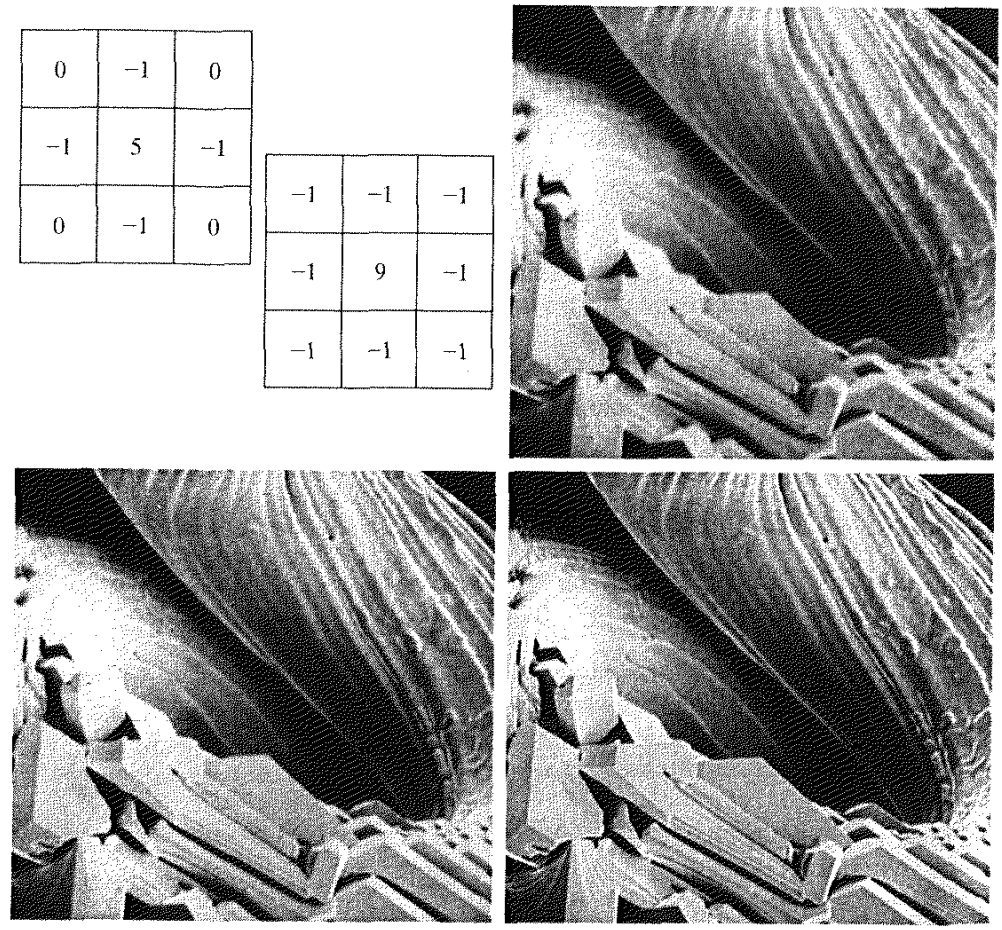
$$\begin{aligned} g(x, y) &= f(x, y) - [f(x + 1, y) + f(x - 1, y) \\ &\quad + f(x, y + 1) + f(x, y - 1)] + 4f(x, y) \\ &= 5f(x, y) - [f(x + 1, y) + f(x - 1, y) \\ &\quad + f(x, y + 1) + f(x, y - 1)]. \end{aligned} \tag{3.7-6}$$

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).

EXAMPLE 3.12: Image enhancement using a composite Laplacian mask.

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×.)

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.



a b c
d e

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

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

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 darkroom 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

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

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

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

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

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

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

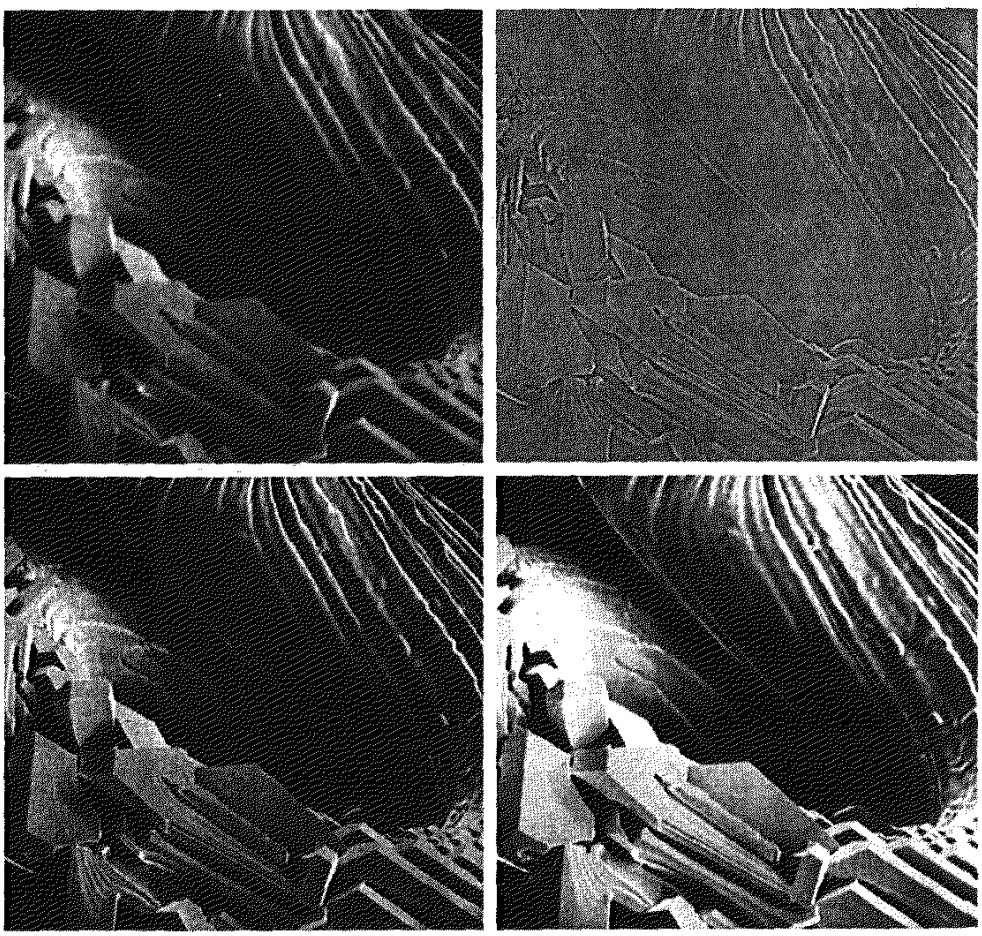
Equation (3.7-10) is applicable in general and does not state explicitly how the sharp image is obtained. If we elect to use the Laplacian, then we know that $f_s(x, y)$ can be obtained using Eq. (3.7-5). In this case, Eq. (3.7-10) becomes

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

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.

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

■ 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.



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*

$$\nabla \mathbf{f} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f}{\partial x} \\ \dfrac{\partial f}{\partial y} \end{bmatrix} \tag{3.7-12}$$

The magnitude of this vector is given by

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

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:

$$\nabla f \approx |G_x| + |G_y|. \tag{3.7-14}$$

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°. 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 3×3 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:

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

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

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

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:

$$\nabla f \approx |z_9 - z_5| + |z_8 - z_6|. \tag{3.7-17}$$

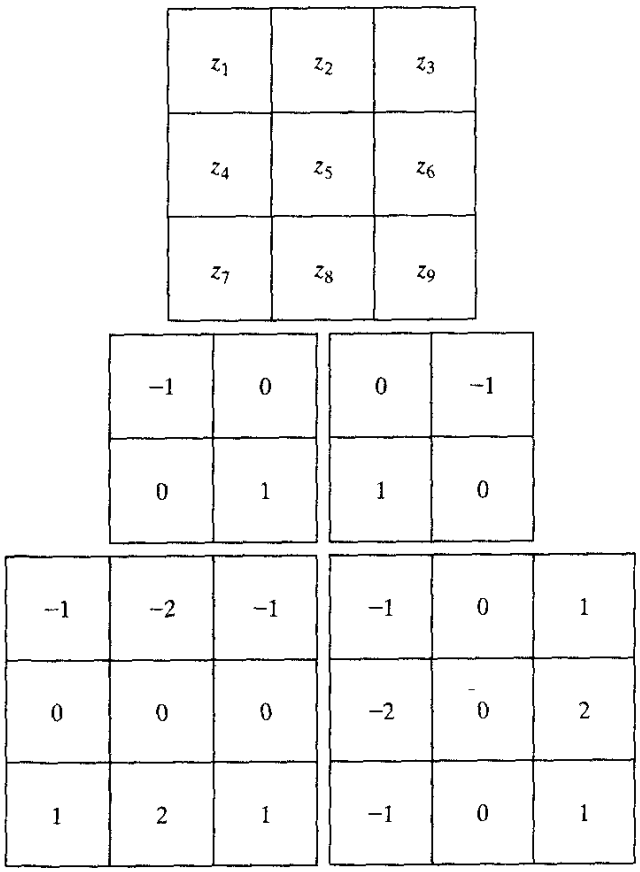
This equation can be implemented with the two masks shown in Figs. 3.44(b) and (c). These masks are referred to as the *Roberts cross-gradient operators*.

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

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

a
b c
d e

FIGURE 3.44
A 3 × 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.

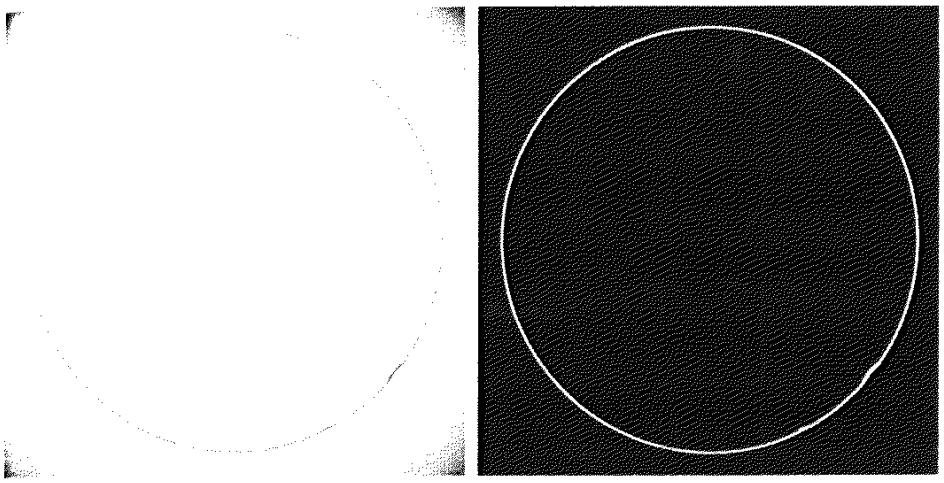


The difference between the third and first rows of the 3 × 3 image region approximates the derivative in the x-direction, and the difference between the third and first columns approximates the derivative in the y-direction. The masks shown in Figs. 3.44(d) and (e), called the *Sobel operators*, can 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.

3.8 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

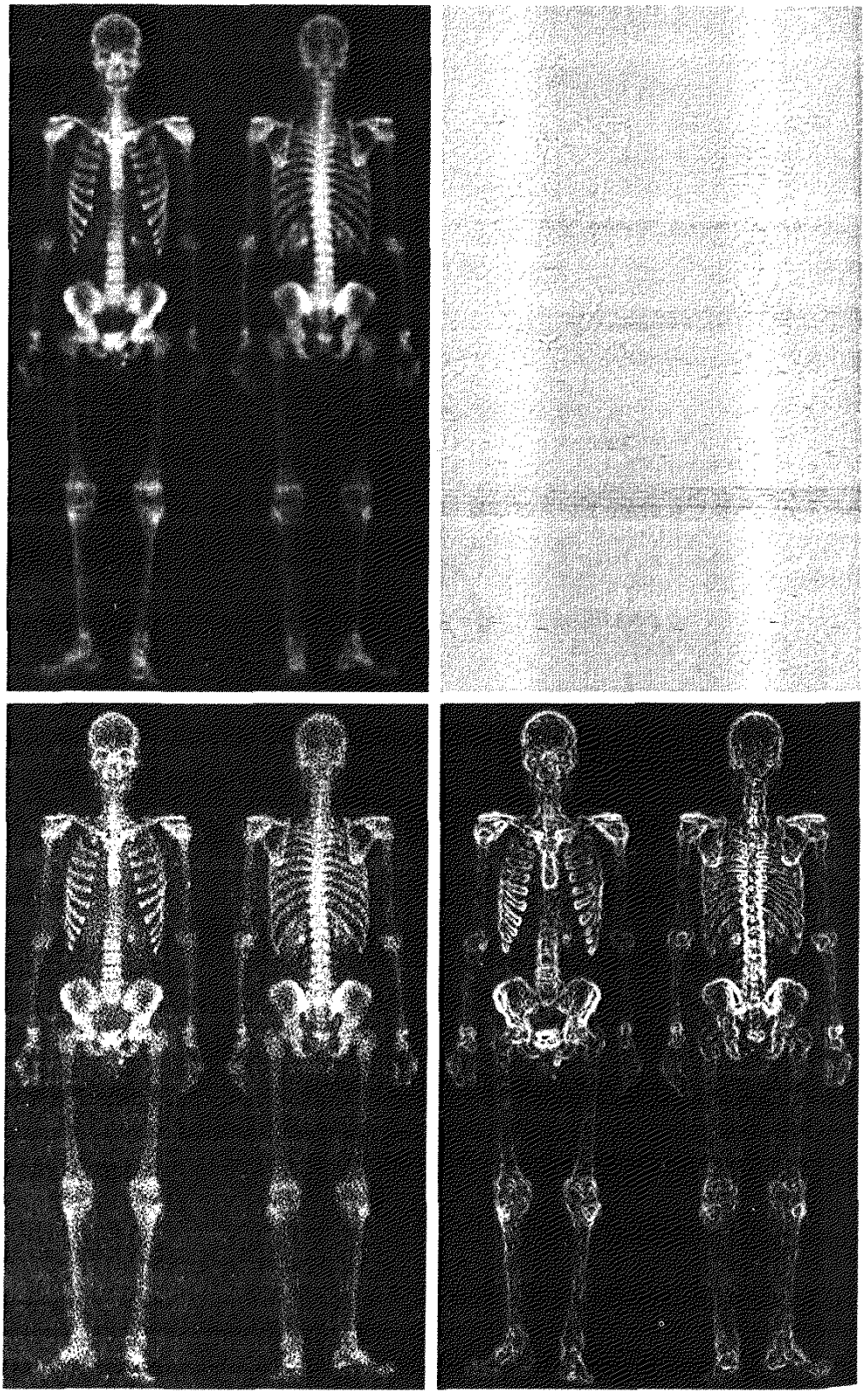
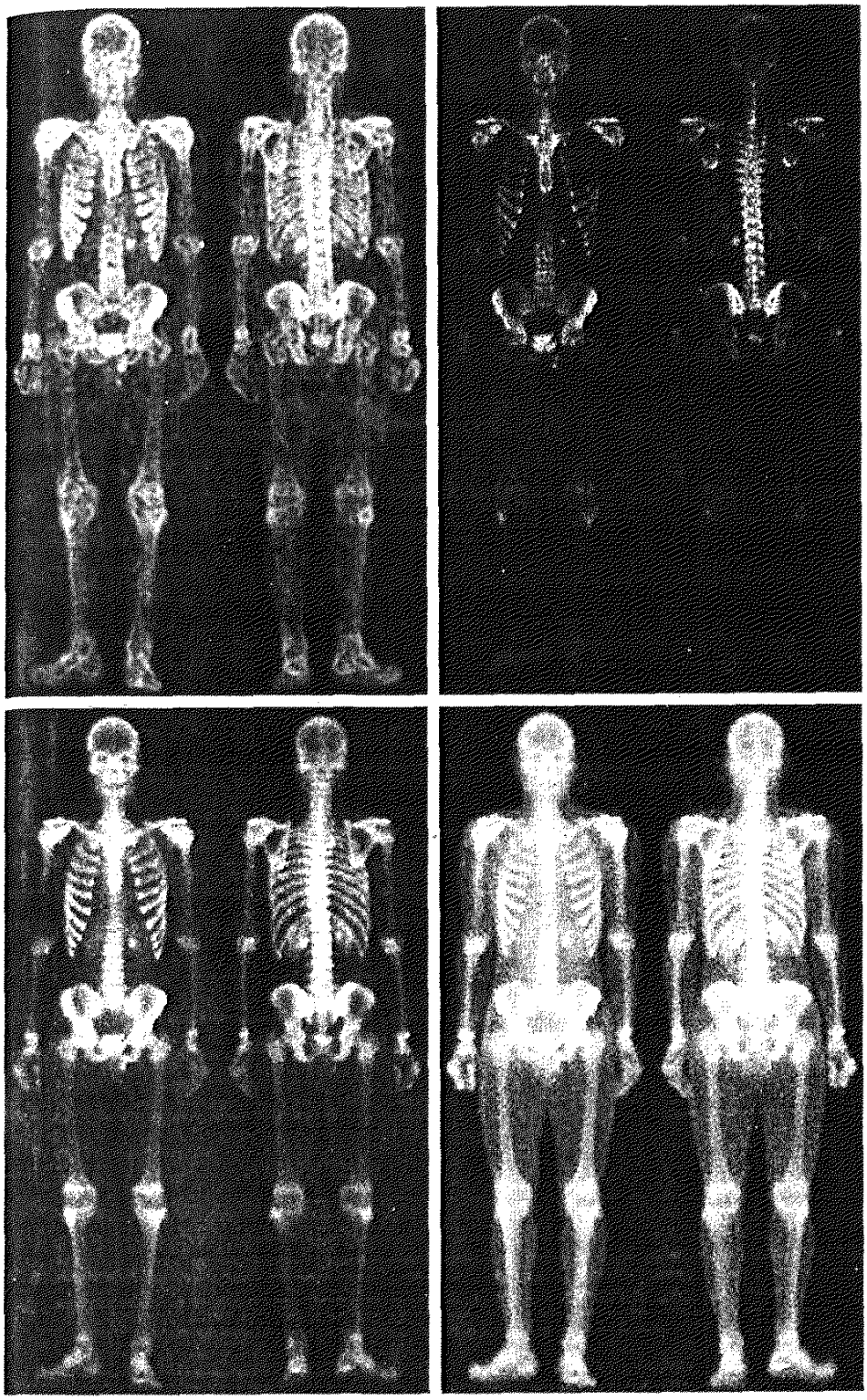


FIGURE 3.46
(a) Image of whole body bone scan. (b) Laplacian of (a). (c) Sharpened image obtained by adding (a) and (b). (d) Sobel of (a).



e f
g h

FIGURE 3.46
(Continued)
(e) Sobel image smoothed with a 5×5 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 nonlinear 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 5×5. 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 γ 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.

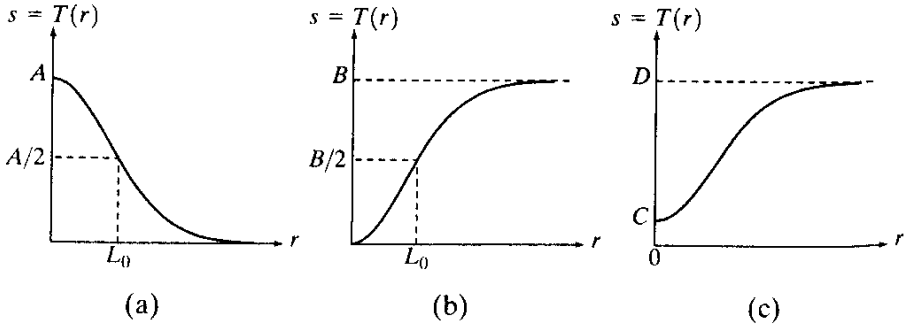
For additional reading on linear spatial filters and their implementation, see Umbaugh [1998], Jain [1989], and Rosenfeld and Kak [1982]. Rank-order filters are discussed in these references as well. Wilburn [1998] discusses generalizations of rank-order filters. The book by Pitas and Venetsanopoulos [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

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.

3.1 Exponentials of the form e^{-ar^2}, with a 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.)



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 $T(r) = 255$ 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 σ are the mean and standard deviation of the Gaussian PDF. The approach is to let m and σ be measures of average gray level and contrast of a given image. What is the transformation function you would use for histogram equalization?

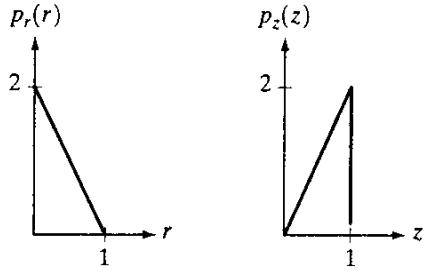
★**3.8** Assuming continuous values, show by example that it is possible to have a case in which the transformation function given in Eq. (3.3-4) satisfies Conditions (a) and (b) in Section 3.3.1, but its inverse may fail to be single valued.

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 $r_k, k = 0, 1, \ldots, L - 1$, 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 3×3 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.

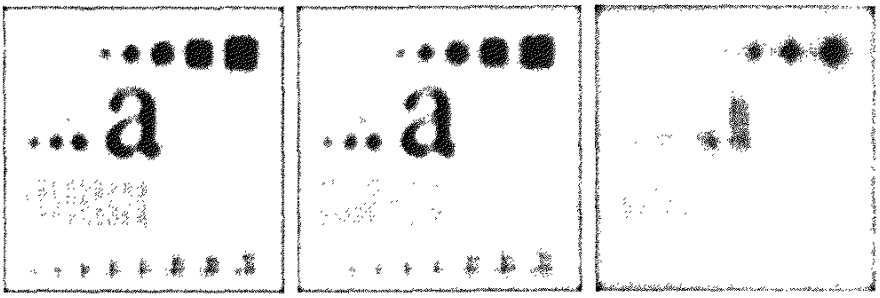
(b) Propose a technique for updating the median as the center of the neighborhood is moved from pixel to pixel.

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.



(a) (b) (c)

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 θ:

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

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

3.26 Give a 3×3 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.