3 Intensity Transformations and Spatial Filtering

It makes all the difference whether one sees darkness through the light or brightness through the shadows.

David Lindsay

Preview

The term *spatial domain* refers to the image plane itself, and image processing methods in this category are based on direct manipulation of pixels in an image. This is in contrast to image processing in a *transform domain* which, as introduced in Section 2.6.7 and discussed in more detail in Chapter 4, involves first transforming an image into the transform domain, doing the processing there, and obtaining the inverse transform to bring the results back into the spatial domain. Two principal categories of spatial processing are intensity transformations and spatial filtering. As you will learn in this chapter, intensity transformations operate on single pixels of an image, principally for the purpose of contrast manipulation and image thresholding. Spatial filtering deals with performing operations, such as image sharpening, by working in a neighborhood of every pixel in an image. In the sections that follow, we discuss a number of "classical" techniques for intensity transformations and spatial filtering. We also discuss in some detail fuzzy techniques that allow us to incorporate imprecise, knowledge-based information in the formulation of intensity transformations and spatial filtering algorithms.

3.1 Background

3.1.1 The Basics of Intensity Transformations and Spatial Filtering

All the image processing techniques discussed in this section are implemented in the *spatial domain*, which we know from the discussion in Section 2.4.2 is simply the plane containing the pixels of an image. As noted in Section 2.6.7, spatial domain techniques operate directly on the pixels of an image as opposed, for example, to the frequency domain (the topic of Chapter 4) in which operations are performed on the Fourier transform of an image, rather than on the image itself. As you will learn in progressing through the book, some image processing tasks are easier or more meaningful to implement in the spatial domain while others are best suited for other approaches. Generally, spatial domain techniques are more efficient computationally and require less processing resources to implement.

The spatial domain processes we discuss in this chapter can 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 output image, and T is an operator on f defined over a neighborhood of point (x, y). The operator can apply to a single image (our principal focus in this chapter) or to a set of images, such as performing the pixel-by-pixel sum of a sequence of images for noise reduction, as discussed in Section 2.6.3. Figure 3.1 shows the basic implementation of Eq. (3.1-1) on a single image. The point (x, y) shown is an arbitrary location in the image, and the small region shown containing the point is a neighborhood of (x, y), as explained in Section 2.6.5. Typically, the neighborhood is rectangular, centered on (x, y), and much smaller in size than the image.

Other neighborhood shapes, such as digital approximations to circles, are used sometimes, but rectangular shapes are by far the most prevalent because they are much easier to implement computationally.

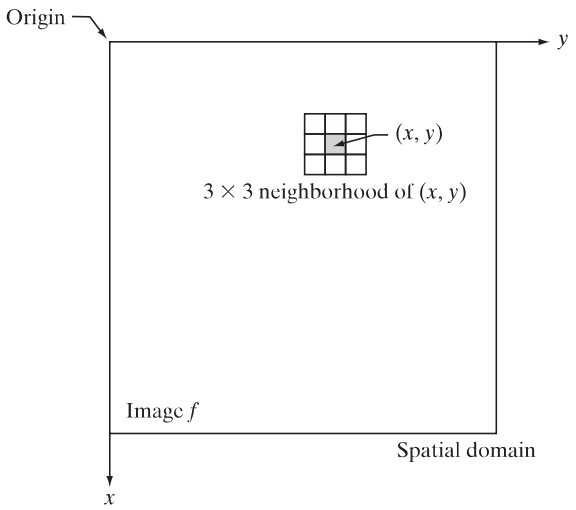


FIGURE 3.1 A 3×3 neighborhood about a point (x, y) in an image in the spatial domain. The neighborhood is moved from pixel to pixel in the image to generate an output image.

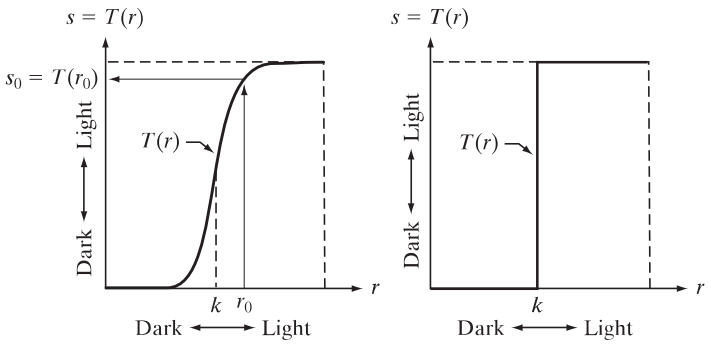
The process that Fig. 3.1 illustrates consists of moving the origin of the neighborhood from pixel to pixel and applying the operator T to the pixels in the neighborhood to yield the output at that location. Thus, for any specific location (x, y), the value of the output image g at those coordinates is equal to the result of applying T to the neighborhood with origin at (x, y) in f. For example, suppose that the neighborhood is a square of size 3×3, and that operator T is defined as "compute the average intensity of the neighborhood." Consider an arbitrary location in an image, say (100, 150). Assuming that the origin of the neighborhood is at its center, the result, $g(100, 150)$, at that location is computed as the sum of $f(100, 150)$ and its 8-neighbors, divided by 9 (i.e., the average intensity of the pixels encompassed by the neighborhood). The origin of the neighborhood is then moved to the next location and the procedure is repeated to generate the next value of the output image g. Typically, the process starts at the top left of the input image and proceeds pixel by pixel in a horizontal scan, one row at a time. When the origin of the neighborhood is at the border of the image, part of the neighborhood will reside outside the image. The procedure is either to ignore the outside neighbors in the computations specified by T, or to pad the image with a border of 0s or some other specified intensity values. The thickness of the padded border depends on the size of the neighborhood. We will return to this issue in Section 3.4.1.

As we discuss in detail in Section 3.4, the procedure just described is called *spatial filtering*, in which the neighborhood, along with a predefined operation, is called a *spatial filter* (also referred to as a *spatial mask*, *kernel*, *template*, or *window*). The type of operation performed in the neighborhood determines the nature of the filtering process.

The smallest possible neighborhood is of size 1×1. In this case, g depends only on the value of f at a single point (x, y) and T in Eq. (3.1-1) becomes an *intensity* (also called *gray-level* or *mapping*) *transformation function* of the form

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

where, for simplicity in notation, s and r are variables denoting, respectively, the intensity of g and f at any point (x, y). For example, if $T(r)$ has the form in Fig. 3.2(a), the effect of applying the transformation to every pixel of f to generate the corresponding pixels in g would be to produce an image of



a b

FIGURE 3.2 Intensity transformation functions. (a) Contrast-stretching function. (b) Thresholding function.

higher contrast than the original by darkening the intensity levels below k and brightening the levels above k. In this technique, sometimes called *contrast stretching* (see Section 3.2.4), values of r lower than k are compressed by the transformation function into a narrow range of s, toward black. The opposite is true for values of r higher than k. Observe how an intensity value r_0 is mapped to obtain the corresponding value s_0. 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 intensity transformation functions. In this chapter, we use intensity transformations principally for image enhancement. In Chapter 10, we use them for image segmentation. Approaches whose results depend only on the intensity at a point sometimes are called *point processing* techniques, as opposed to the *neighborhood processing* techniques discussed earlier in this section.

3.1.2 About the Examples in This Chapter

Although intensity transformations and spatial filtering span a broad range of applications, most of the examples in this chapter are applications to image enhancement. *Enhancement* is the process of manipulating an image so that the result is more suitable than the original for a specific application. The word *specific* is important here because it establishes at the outset that enhancement techniques are problem oriented. Thus, for example, a method that is quite useful for enhancing X-ray images may not be the best approach for enhancing satellite images taken in the infrared band of the electromagnetic spectrum. 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. When dealing with machine perception, a given technique is easier to quantify. For example, in an automated character-recognition system, the most appropriate enhancement method is the one that results in the best recognition rate, leaving aside other considerations such as computational requirements of one method over another.

Regardless of the application or method used, however, image enhancement is one of the most visually appealing areas of image processing. By its very nature, beginners in image processing generally find enhancement applications interesting and relatively simple to understand. Therefore, using examples from image enhancement to illustrate the spatial processing methods developed in this chapter not only saves having an extra chapter in the book dealing with image enhancement but, more importantly, is an effective approach for introducing newcomers to the details of processing techniques in the spatial domain. As you will see as you progress through the book, the basic material developed in this chapter is applicable to a much broader scope than just image enhancement.

3.2 Some Basic Intensity Transformation Functions

Intensity transformations are among the simplest of all image processing 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. Because we are dealing with digital quantities, values of a 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 intensity 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 intensity 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

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

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

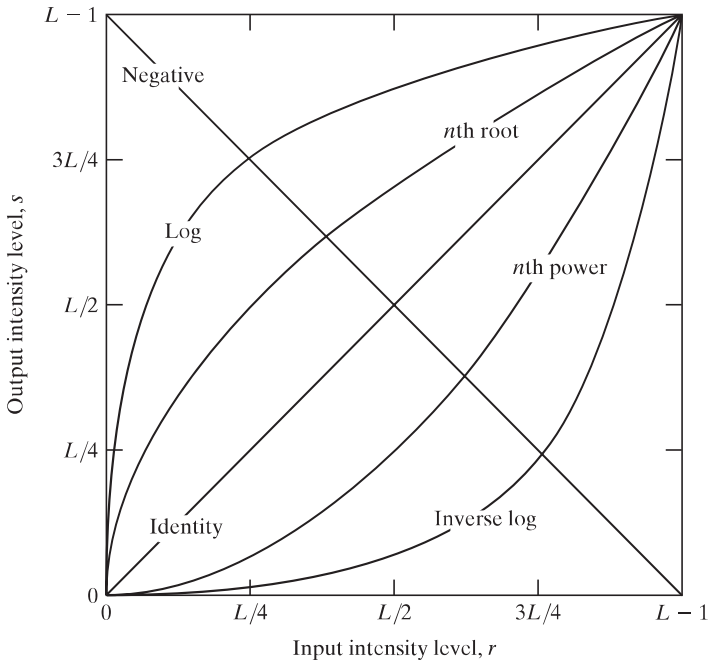
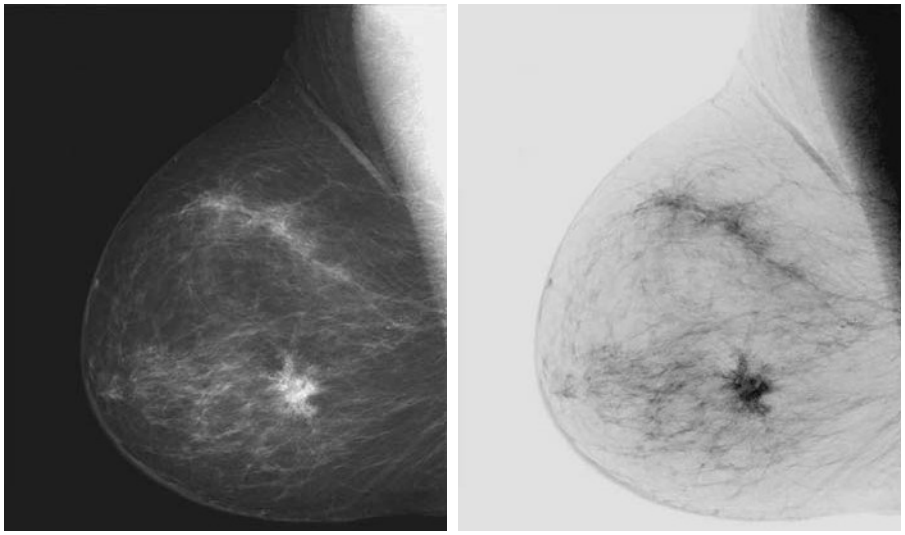


FIGURE 3.3 Some basic intensity transformation functions. All curves were scaled to fit in the range shown.



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.)

image, especially when the black areas are dominant in size. Figure 3.4 shows an example. 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 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 intensity values in the input into a wider range of output levels. The opposite is true of higher values of input levels. We 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 intensity levels in an image, but the power-law transformations discussed in the next section are much more versatile for this purpose. 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 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

a b

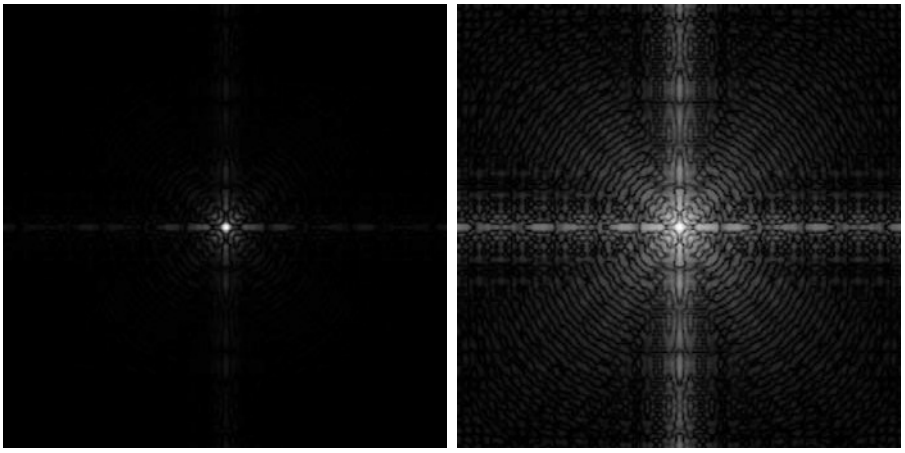


FIGURE 3.5
(a) Fourier spectrum. (b) Result of applying the log transformation in Eq. (3.2-2) with $c = 1$.

faithfully such a wide range of intensity values. The net effect is that a significant degree of intensity detail can 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 becomes 0 to 6.2, which is more manageable. 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 an unmodified 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.

3.2.3 Power-Law (Gamma) 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 higher values of input levels. Unlike the log function, however, we notice

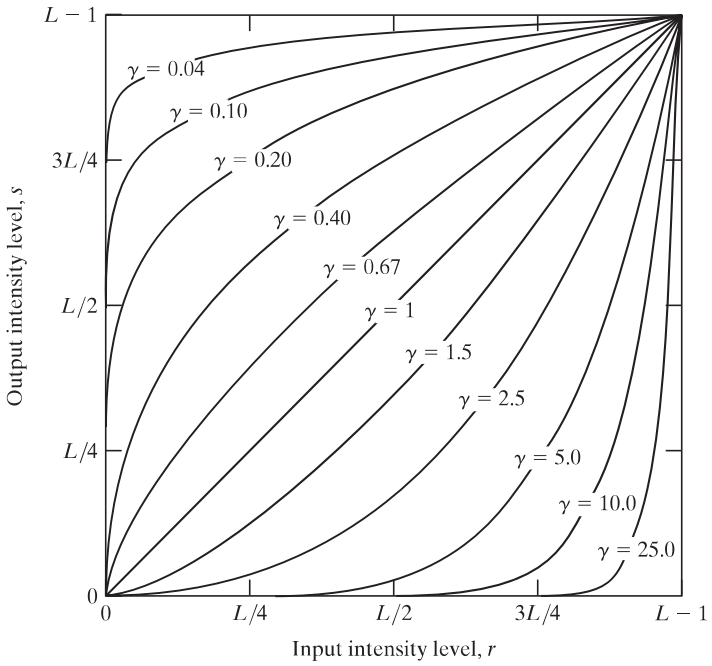


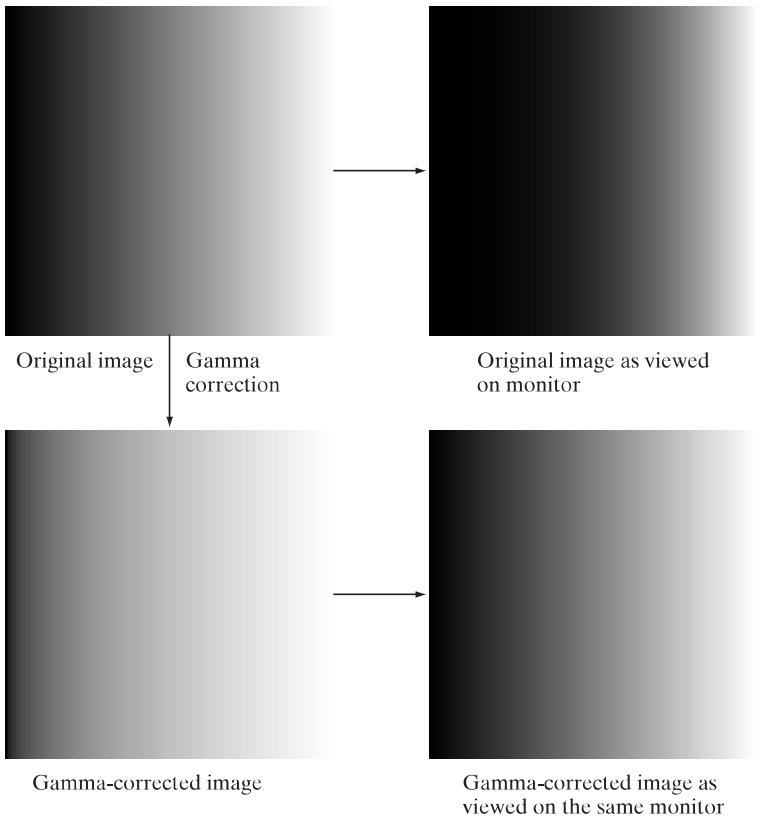
FIGURE 3.6 Plots of the equation $s = cr^{\gamma}$ for various values of γ ($c = 1$ in all cases). All curves were scaled to fit in the range shown.

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 these 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 intensity-ramp image input into a monitor. As expected, the output of the monitor appears darker than the input, as Fig. 3.7(b) shows. 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 Fig. 3.7(d) shows. A similar analysis would apply to other imaging devices such as scanners and printers. The only difference would be the device-dependent value of gamma (Poynton [1996]).

a b
c d

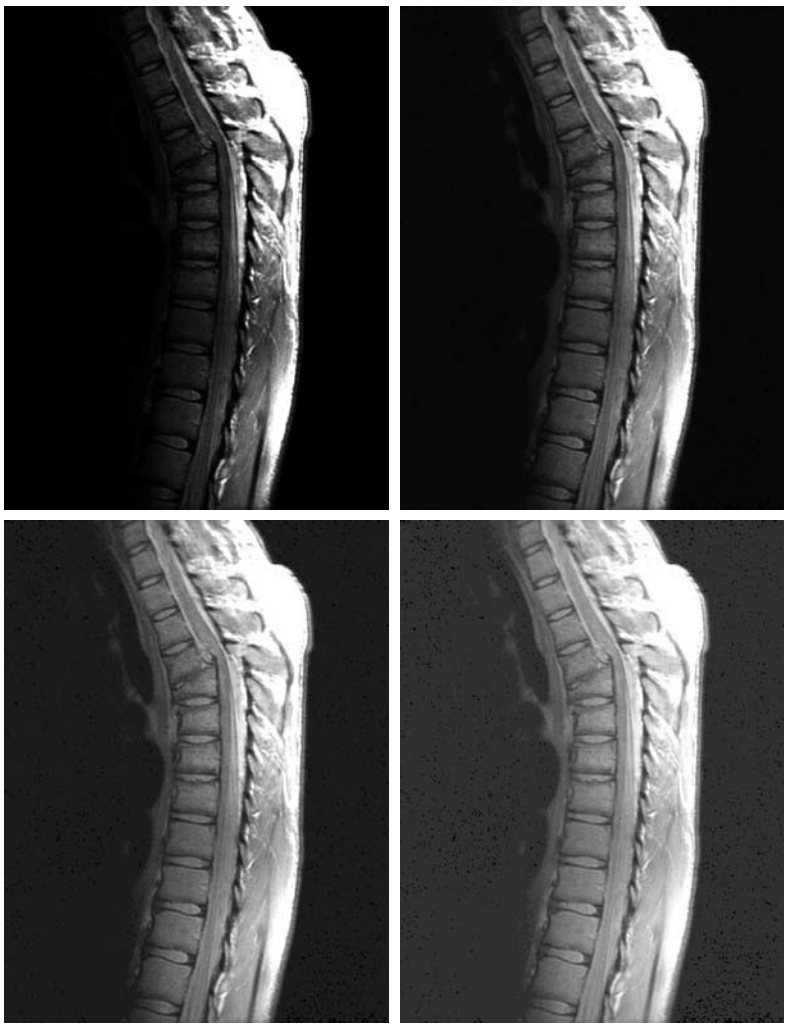
FIGURE 3.7
(a) Intensity ramp image. (b) Image as viewed on a simulated monitor with a gamma of 2.5. (c) Gamma-corrected image. (d) Corrected image as viewed on the same monitor. Compare (d) and (a).



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 changes not only the intensity, but also the ratios of red to green to blue in a color image. Gamma correction has become increasingly important in the past few years, as the 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.

■ In addition to gamma correction, power-law transformations are useful for general-purpose contrast manipulation. Figure 3.8(a) shows a magnetic resonance image (MRI) of an upper thoracic human spine with a fracture dislocation 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. Because the given image is predominantly dark, an expansion of intensity levels is 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

EXAMPLE 3.1: Contrast enhancement using power-law transformations.



a b
c d

FIGURE 3.8 (a) Magnetic resonance image (MRI) 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 courtesy of Dr. David R. Pickens, Department of Radiology and Radiological Sciences, Vanderbilt University Medical Center.)

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" appearance, 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 processed now has a washed-out appearance, indicating that a compression of intensity 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

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.)

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. ■

3.2.4 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 you 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 the wrong setting of a lens aperture during image acquisition. *Contrast stretching* is a process that expands the range of intensity levels in an image so that it spans the full intensity range of the recording medium or display device.

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 function. If $r_1 = s_1$ and $r_2 = s_2$, the transformation is a linear function that produces no changes in intensity 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 intensity 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 intensity levels, thus preventing the creation of intensity artifacts in the processed image.

Figure 3.10(b) shows an 8-bit image with low contrast. Figure 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 intensity 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 $(r_1, s_1) = (m, 0)$ and $(r_2, s_2) = (m, L - 1)$, where m is the mean intensity 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.

Intensity-level slicing

Highlighting a specific range of intensities in an image often is of interest. Applications include enhancing features such as masses of water in satellite imagery and enhancing flaws in X-ray images. The process, often called *intensity-level*

a b
c d

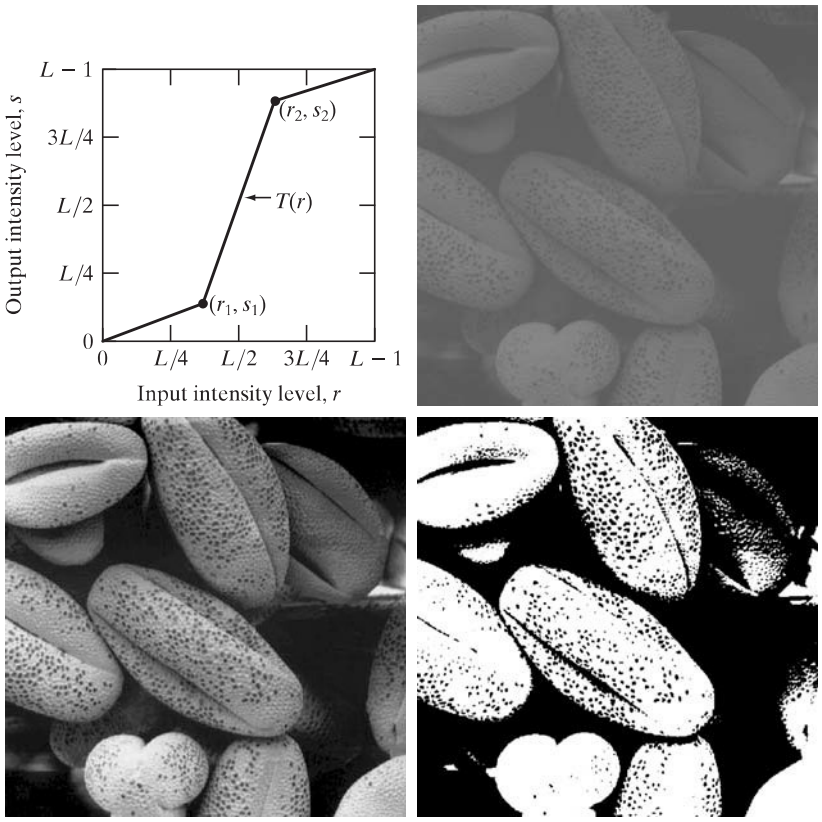


FIGURE 3.10 Contrast stretching. (a) Form of transformation function. (b) A low-contrast image. (c) Result of contrast stretching. (d) Result of thresholding. (Original image courtesy of Dr. Roger Heady, Research School of Biological Sciences, Australian National University, Canberra, Australia.)

slicing, can be implemented in several ways, but most are variations of two basic themes. One approach is to display in one value (say, white) all the values in the range of interest and in another (say, black) all other intensities. This transformation, shown in Fig. 3.11(a), produces a binary image. The second approach, based on the transformation in Fig. 3.11(b), brightens (or darkens) the desired range of intensities but leaves all other intensity levels in the image unchanged.

a b

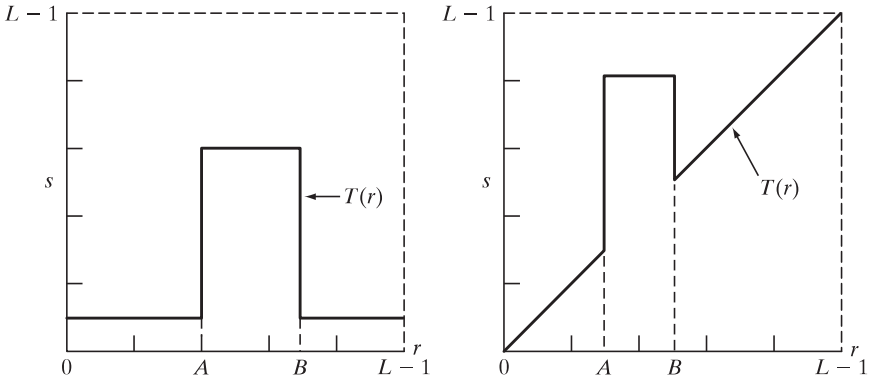


FIGURE 3.11 (a) This transformation highlights intensity range $[A, B]$ and reduces all other intensities to a lower level. (b) This transformation highlights range $[A, B]$ and preserves all other intensity levels.

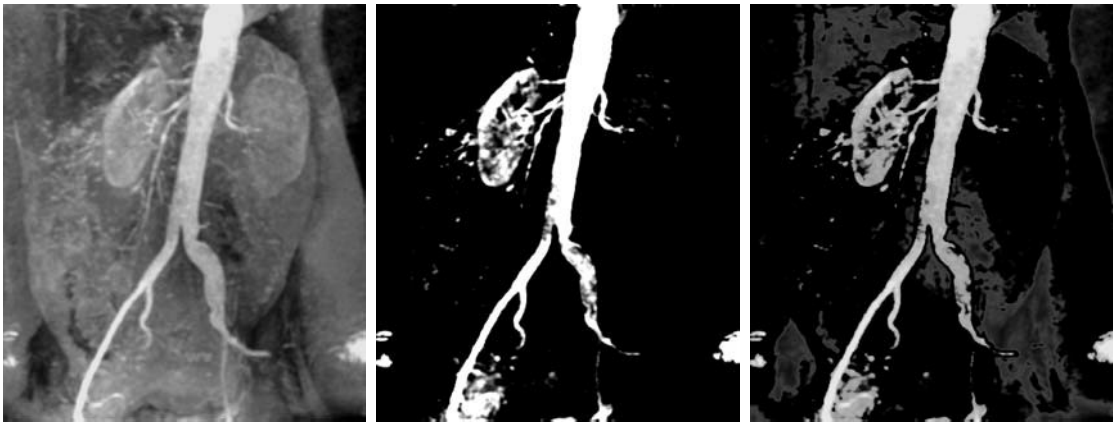
EXAMPLE 3.3: Intensity-level slicing.

■ Figure 3.12(a) is an aortic angiogram near the kidney area (see Section 1.3.2 for a more detailed explanation of this image). The objective of this example is to use intensity-level slicing to highlight the major blood vessels that appear brighter as a result of an injected contrast medium. Figure 3.12(b) shows the result of using a transformation of the form in Fig. 3.11(a), with the selected band near the top of the scale, because the range of interest is brighter than the background. The net result of this transformation is that the blood vessel and parts of the kidneys appear white, while all other intensities are black. This type of enhancement produces a binary image and is useful for studying the *shape* of the flow of the contrast medium (to detect blockages, for example).

If, on the other hand, interest lies in the actual intensity values of the region of interest, we can use the transformation in Fig. 3.11(b). Figure 3.12(c) shows the result of using such a transformation in which a band of intensities in the mid-gray region around the mean intensity was set to black, while all other intensities were left unchanged. Here, we see that the gray-level tonality of the major blood vessels and part of the kidney area were left intact. Such a result might be useful when interest lies in measuring the actual flow of the contrast medium as a function of time in a series of images. ■

Bit-plane slicing

Pixels are digital numbers composed of bits. For example, the intensity of each pixel in a 256-level gray-scale image is composed of 8 bits (i.e., one byte). Instead of highlighting intensity-level ranges, we could highlight the contribution



a b c

FIGURE 3.12 (a) Aortic angiogram. (b) Result of using a slicing transformation of the type illustrated in Fig. 3.11(a), with the range of intensities of interest selected in the upper end of the gray scale. (c) Result of using the transformation in Fig. 3.11(b), with the selected area set to black, so that grays in the area of the blood vessels and kidneys were preserved. (Original image courtesy of Dr. Thomas R. Gest, University of Michigan Medical School.)

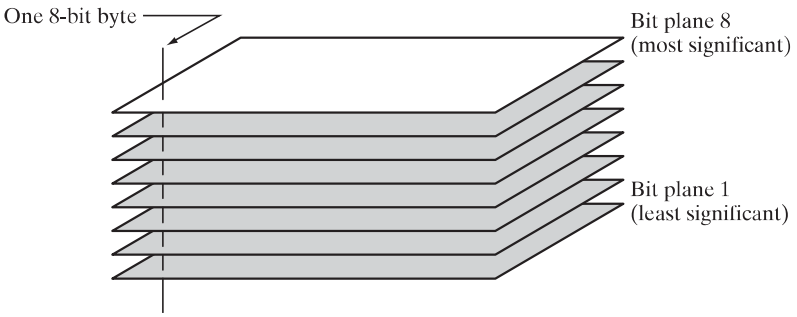
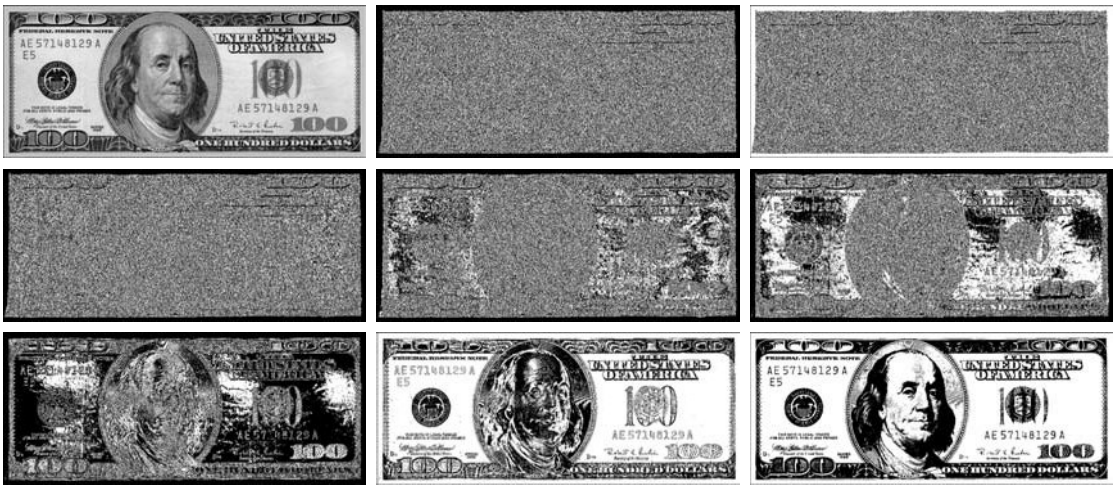


FIGURE 3.13 Bit-plane representation of an 8-bit image.

made to total image appearance by specific bits. As Fig. 3.13 illustrates, an 8-bit image may be considered as being composed of eight 1-bit planes, with plane 1 containing the lowest-order bit of all pixels in the image and plane 8 all the highest-order bits.

Figure 3.14(a) shows an 8-bit gray-scale image and Figs. 3.14(b) through (i) are its eight 1-bit planes, with Fig. 3.14(b) corresponding to the lowest-order bit. Observe that the four higher-order bit planes, especially the last two, contain a significant amount of the visually significant data. The lower-order planes contribute to more subtle intensity details in the image. The original image has a gray border whose intensity is 194. Notice that the corresponding borders of some of the bit planes are black (0), while others are white (1). To see why, consider a



a b c
d e f
g h i

FIGURE 3.14 (a) An 8-bit gray-scale image of size 500 × 1192 pixels. (b) through (i) Bit planes 1 through 8, with bit plane 1 corresponding to the least significant bit. Each bit plane is a binary image.

pixel in, say, the middle of the lower border of Fig. 3.14(a). The corresponding pixels in the bit planes, starting with the highest-order plane, have values 1 1 0 0 0 0 1 0, which is the binary representation of decimal 194. The value of any pixel in the original image can be similarly reconstructed from its corresponding binary-valued pixels in the bit planes.

In terms of intensity transformation functions, it is not difficult to show that the binary image for the 8th bit plane of an 8-bit image can be obtained by processing the input image with a thresholding intensity transformation function that maps all intensities between 0 and 127 to 0 and maps all levels between 128 and 255 to 1. The binary image in Fig. 3.14(i) was obtained in just this manner. It is left as an exercise (Problem 3.4) to obtain the intensity transformation functions for generating the other bit planes.

Decomposing an image into its bit planes is useful for analyzing the relative importance of each bit in the image, a process that aids in determining the adequacy of the number of bits used to quantize the image. Also, this type of decomposition is useful for image compression (the topic of Chapter 8), in which fewer than all planes are used in reconstructing an image. For example, Fig. 3.15(a) shows an image reconstructed using bit planes 8 and 7. The reconstruction is done by multiplying the pixels of the nth plane by the constant 2^{n-1}. This is nothing more than converting the nth significant binary bit to decimal. Each plane used is multiplied by the corresponding constant, and all planes used are added to obtain the gray scale image. Thus, to obtain Fig. 3.15(a), we multiplied bit plane 8 by 128, bit plane 7 by 64, and added the two planes. Although the main features of the original image were restored, the reconstructed image appears flat, especially in the background. This is not surprising because two planes can produce only four distinct intensity levels. Adding plane 6 to the reconstruction helped the situation, as Fig. 3.15(b) shows. Note that the background of this image has perceptible false contouring. This effect is reduced significantly by adding the 5th plane to the reconstruction, as Fig. 3.15(c) illustrates. Using more planes in the reconstruction would not contribute significantly to the appearance of this image. Thus, we conclude that storing the four highest-order bit planes would allow us to reconstruct the original image in acceptable detail. Storing these four planes instead of the original image requires 50% less storage (ignoring memory architecture issues).

a b c

FIGURE 3.15 Images reconstructed using (a) bit planes 8 and 7; (b) bit planes 8, 7, and 6; and (c) bit planes 8, 7, 6, and 5. Compare (c) with Fig. 3.14(a).

3.3 Histogram Processing

The *histogram* of a digital image with intensity levels in the range $[0, L-1]$ is a discrete function $h(r_k) = n_k$, where r_k is the kth intensity value and n_k is the number of pixels in the image with intensity r_k. It is common practice to normalize a histogram by dividing each of its components by the total number of pixels in the image, denoted by the product MN, where, as usual, M and N are the row and column dimensions of the image. Thus, a normalized histogram is given by $p(r_k) = r_k/MN$, for $k = 0, 1, 2, \ldots, L-1$. Loosely speaking, $p(r_k)$ is an estimate of the probability of occurrence of intensity level r_k in an image. The sum of all components of a normalized histogram is equal to 1.

Consult the book Web site for a review of basic probability theory.

Histograms are the basis for numerous spatial domain processing techniques. Histogram manipulation can be used 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 histogram processing for intensity transformations, consider Fig. 3.16, which is the pollen image of Fig. 3.10 shown in four basic intensity 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 intensity values, r_k. The vertical axis corresponds to values of $h(r_k) = n_k$ or $p(r_k) = n_k/MN$ if the values are normalized. Thus, histograms may be viewed graphically simply as plots of $h(r_k) = n_k$ versus r_k or $p(r_k) = n_k/MN$ versus r_k.

We note in the dark image that the components of the histogram are concentrated on the low (dark) side of the intensity scale. Similarly, the components of the histogram of the light image are biased toward the high side of the scale. An image with low contrast has a narrow histogram located typically toward the middle of the intensity 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 wide range of the intensity 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 intensity 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.

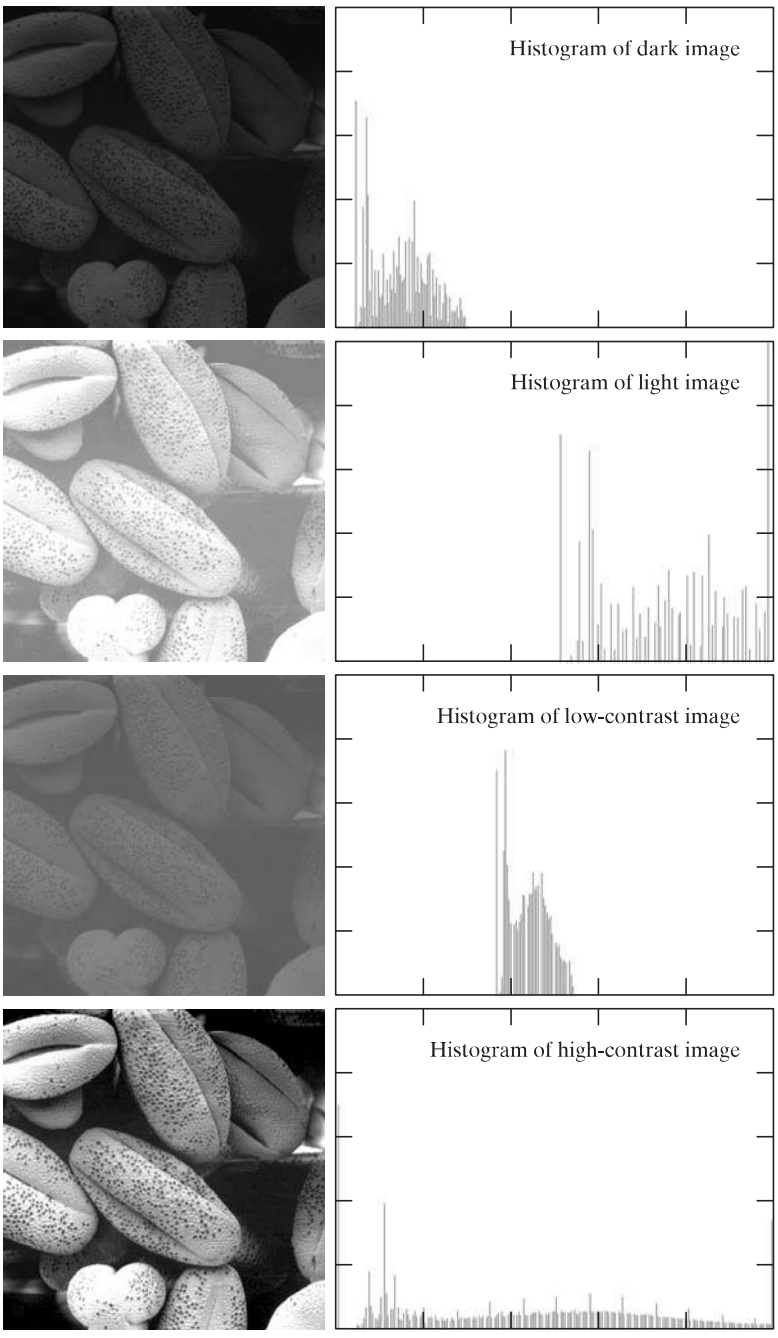


FIGURE 3.16 Four basic image types: dark, light, low contrast, high contrast, and their corresponding histograms.

3.3.1 Histogram Equalization

Consider for a moment continuous intensity values and let the variable r denote the intensities of an image to be processed. As usual, we assume that r is in the range $[0, L-1]$, with $r = 0$ representing black and $r = L-1$ representing white. For r satisfying these conditions, we focus attention on transformations (intensity mappings) of the form

$$s = T(r) \quad 0 \le r \le L-1 \tag{3.3-1}$$

that produce an output intensity level s for every pixel in the input image having intensity r. We assume that:

(a) $T(r)$ is a monotonically[†] increasing function in the interval $0 \le r \le L-1$; and

(b) $0 \le T(r) \le L-1$ for $0 \le r \le L-1$.

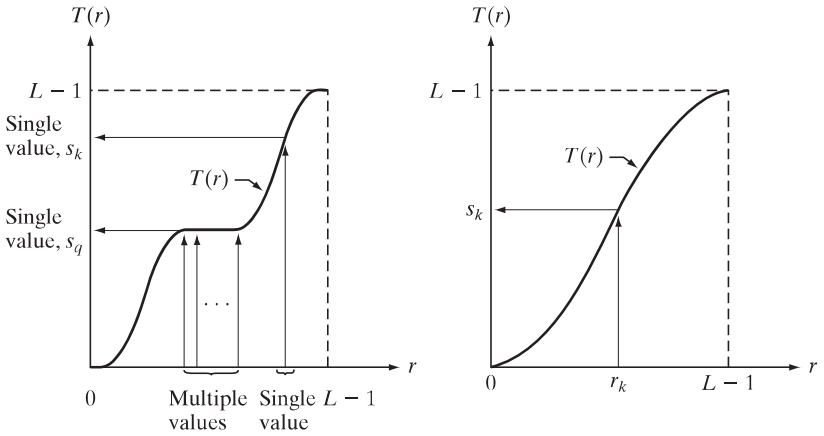
In some formulations to be discussed later, we use the inverse

$$r = T^{-1}(s) \quad 0 \le s \le L-1 \tag{3.3-2}$$

in which case we change condition (a) to

(a′) $T(r)$ is a strictly monotonically increasing function in the interval $0 \le r \le L-1$.

The requirement in condition (a) that $T(r)$ be monotonically increasing guarantees that output intensity values will never be less than corresponding input values, thus preventing artifacts created by reversals of intensity. Condition (b) guarantees that the range of output intensities is the same as the input. Finally, condition (a′) guarantees that the mappings from s back to r will be one-to-one, thus preventing ambiguities. Figure 3.17(a) shows a function



a b

FIGURE 3.17 (a) Monotonically increasing function, showing how multiple values can map to a single value. (b) Strictly monotonically increasing function. This is a one-to-one mapping, both ways.

[†]Recall that a function $T(r)$ is *monotonically increasing* if $T(r_2) \ge T(r_1)$ for $r_2 > r_1$. $T(r)$ is a *strictly monotonically increasing* function if $T(r_2) > T(r_1)$ for $r_2 > r_1$. Similar definitions apply to monotonically decreasing functions.

that satisfies conditions (a) and (b). Here, we see that it is possible for multiple values to map to a single value and still satisfy these two conditions. That is, a monotonic transformation function performs a one-to-one or many-to-one mapping. This is perfectly fine when mapping from r to s. However, Fig. 3.17(a) presents a problem if we wanted to recover the values of r uniquely from the mapped values (inverse mapping can be visualized by reversing the direction of the arrows). This would be possible for the inverse mapping of s_k in Fig. 3.17(a), but the inverse mapping of s_q is a *range* of values, which, of course, prevents us in general from recovering the original value of r that resulted in s_q. As Fig. 3.17(b) shows, requiring that $T(r)$ be strictly monotonic guarantees that the inverse mappings will be *single valued* (i.e., the mapping is one-to-one in both directions). This is a theoretical requirement that allows us to derive some important histogram processing techniques later in this chapter. Because in practice we deal with integer intensity values, we are forced to round all results to their nearest integer values. Therefore, when strict monotonicity is not satisfied, we address the problem of a nonunique inverse transformation by looking for the closest integer matches. Example 3.8 gives an illustration of this.

The intensity levels in an image may be viewed as random variables in the interval $[0, L-1]$. A fundamental descriptor of a random variable is its probability density function (PDF). Let $p_r(r)$ and $p_s(s)$ denote the PDFs of r and s, respectively, where the subscripts on p are used to indicate that p_r and p_s are different functions in general. A fundamental result from basic probability theory is that if $p_r(r)$ and $T(r)$ are known, and $T(r)$ is continuous and differentiable over the range of values of interest, then the PDF of the transformed (mapped) variable s can be obtained using the simple formula

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

Thus, we see that the PDF of the output intensity variable, s, is determined by the PDF of the input intensities and the transformation function used [recall that r and s are related by $T(r)$].

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

$$s = T(r) = (L-1)\int_0^r p_r(w)\,dw \tag{3.3-4}$$

where w is a dummy variable of integration. The right side of this equation is recognized as the cumulative distribution function (CDF) of random variable r. Because PDFs always are positive, and recalling that the integral of a function is the area under the function, it follows that the transformation function of Eq. (3.3-4) satisfies condition (a) because the area under the function cannot decrease as r increases. When the upper limit in this equation is $r = (L-1)$, the integral evaluates to 1 (the area under a PDF curve always is 1), so the maximum value of s is $(L-1)$ and condition (b) is satisfied also.

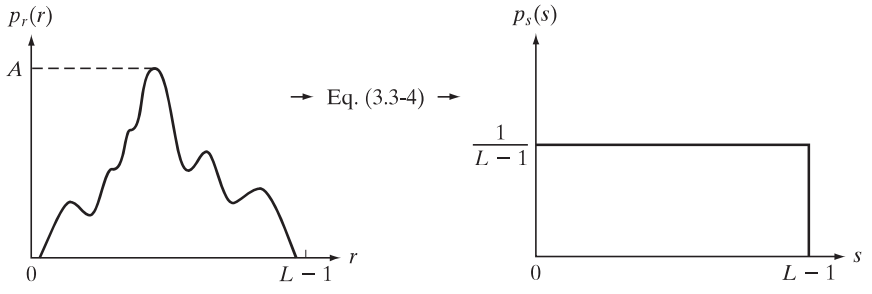
To find the $p_s(s)$ corresponding to the transformation just discussed, we use Eq. (3.3-3). We know from Leibniz's rule in basic calculus that the derivative of a definite integral with respect to its upper limit is the integrand evaluated at the limit. That is,

$$\begin{aligned}\frac{ds}{dr} &= \frac{dT(r)}{dr} \\ &= (L-1)\frac{d}{dr}\left[\int_0^r p_r(w)\,dw\right] \\ &= (L-1)p_r(r)\end{aligned} \tag{3.3-5}$$

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

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

We recognize the form of $p_s(s)$ in the last line of this equation as a *uniform* probability density function. Simply stated, we have demonstrated that performing the intensity transformation in Eq. (3.3-4) yields a random variable, s, characterized by a uniform PDF. It is important to note from this equation that $T(r)$ depends on $p_r(r)$ but, as Eq. (3.3-6) shows, the resulting $p_s(s)$ *always* is uniform, *independently* of the form of $p_r(r)$. Figure 3.18 illustrates these concepts.



a b

FIGURE 3.18 (a) An arbitrary PDF. (b) Result of applying the transformation in Eq. (3.3-4) to all intensity levels, r. The resulting intensities, s, have a uniform PDF, independently of the form of the PDF of the r's.

EXAMPLE 3.4: Illustration of Eqs. (3.3-4) and (3.3-6).

■ To fix ideas, consider the following simple example. Suppose that the (continuous) intensity values in an image have the PDF

$$p_r(r) = \begin{cases} \dfrac{2r}{(L-1)^2} & \text{for } 0 \le r \le L-1 \\ 0 & \text{otherwise} \end{cases}$$

From Eq. (3.3-4),

$$s = T(r) = (L-1)\int_0^r p_r(w)\,dw = \frac{2}{L-1}\int_0^r w\,dw = \frac{r^2}{L-1}$$

Suppose next that we form a new image with intensities, s, obtained using this transformation; that is, the s values are formed by squaring the corresponding intensity values of the input image and dividing them by $(L-1)$. For example, consider an image in which $L = 10$, and suppose that a pixel in an arbitrary location (x, y) in the input image has intensity $r = 3$. Then the pixel in that location in the new image is $s = T(r) = r^2/9 = 1$. We can verify that the PDF of the intensities in the new image is uniform simply by substituting $p_r(r)$ into Eq. (3.3-6) and using the fact that $s = r^2/(L-1)$; that is,

$$\begin{aligned} p_s(s) &= p_r(r)\left|\frac{dr}{ds}\right| = \frac{2r}{(L-1)^2}\left|\left[\frac{ds}{dr}\right]^{-1}\right| \\ &= \frac{2r}{(L-1)^2}\left|\left[\frac{d}{dr}\frac{r^2}{L-1}\right]^{-1}\right| \\ &= \frac{2r}{(L-1)^2}\left|\frac{(L-1)}{2r}\right| = \frac{1}{L-1} \end{aligned}$$

where the last step follows from the fact that r is nonnegative and we assume that $L > 1$. As expected, the result is a uniform PDF. ■

For discrete values, we deal with probabilities (histogram values) and summations instead of probability density functions and integrals.[†] As mentioned earlier, the probability of occurrence of intensity level r_k in a digital image is approximated by

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

where MN is the total number of pixels in the image, n_k is the number of pixels that have intensity r_k, and L is the number of possible intensity levels in the image (e.g., 256 for an 8-bit image). As noted in the beginning of this section, a plot of $p_r(r_k)$ versus r_k is commonly referred to as a *histogram*.

[†]The conditions of monotonicity stated earlier apply also in the discrete case. We simply restrict the values of the variables to be discrete.

The discrete form of the transformation in Eq. (3.3-4) is

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

Thus, a processed (output) image is obtained by mapping each pixel in the input image with intensity r_k into a corresponding pixel with level s_k in the output image, using Eq. (3.3-8). The transformation (mapping) $T(r_k)$ in this equation is called a *histogram equalization* or *histogram linearization* transformation. It is not difficult to show (Problem 3.10) that this transformation satisfies conditions (a) and (b) stated previously in this section.

EXAMPLE 3.5: A simple illustration of histogram equalization.

■ Before continuing, it will be helpful to work through a simple example. Suppose that a 3-bit image ($L = 8$) of size 64×64 pixels ($MN = 4096$) has the intensity distribution shown in Table 3.1, where the intensity levels are integers in the range $[0, L-1] = [0, 7]$.

The histogram of our hypothetical image is sketched in Fig. 3.19(a). Values of the histogram equalization transformation function are obtained using Eq. (3.3-8). For instance,

$$s_0 = T(r_0) = 7\sum_{j=0}^{0} p_r(r_j) = 7p_r(r_0) = 1.33$$

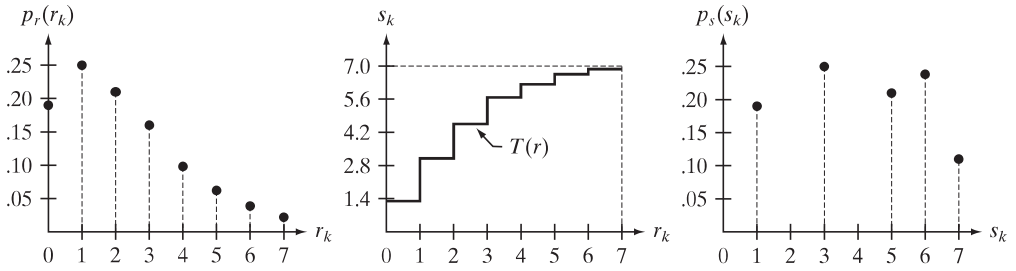
Similarly,

$$s_1 = T(r_1) = 7\sum_{j=0}^{1} p_r(r_j) = 7p_r(r_0) + 7p_r(r_1) = 3.08$$

and $s_2 = 4.55$, $s_3 = 5.67$, $s_4 = 6.23$, $s_5 = 6.65$, $s_6 = 6.86$, $s_7 = 7.00$. This transformation function has the staircase shape shown in Fig. 3.19(b).

TABLE 3.1 Intensity distribution and histogram values for a 3-bit, 64×64 digital image.

r_k	n_k	$p_r(r_k) = n_k/MN$
$r_0 = 0$	790	0.19
$r_1 = 1$	1023	0.25
$r_2 = 2$	850	0.21
$r_3 = 3$	656	0.16
$r_4 = 4$	329	0.08
$r_5 = 5$	245	0.06
$r_6 = 6$	122	0.03
$r_7 = 7$	81	0.02



a b c

FIGURE 3.19 Illustration of histogram equalization of a 3-bit (8 intensity levels) image. (a) Original histogram. (b) Transformation function. (c) Equalized histogram.

At this point, the s values still have fractions because they were generated by summing probability values, so we round them to the nearest integer:

$$s_0 = 1.33 \rightarrow 1 \qquad s_4 = 6.23 \rightarrow 6$$
$$s_1 = 3.08 \rightarrow 3 \qquad s_5 = 6.65 \rightarrow 7$$
$$s_2 = 4.55 \rightarrow 5 \qquad s_6 = 6.86 \rightarrow 7$$
$$s_3 = 5.67 \rightarrow 6 \qquad s_7 = 7.00 \rightarrow 7$$

These are the values of the equalized histogram. Observe that there are only five distinct intensity levels. Because $r_0 = 0$ was mapped to $s_0 = 1$, there are 790 pixels in the histogram equalized image with this value (see Table 3.1). Also, there are in this image 1023 pixels with a value of $s_1 = 3$ and 850 pixels with a value of $s_2 = 5$. However both r_3 and r_4 were mapped to the same value, 6, so there are $(656 + 329) = 985$ pixels in the equalized image with this value. Similarly, there are $(245 + 122 + 81) = 448$ pixels with a value of 7 in the histogram equalized image. Dividing these numbers by $MN = 4096$ yielded the equalized histogram in Fig. 3.19(c).

Because a histogram is an approximation to a PDF, and no new allowed intensity levels are created in the process, perfectly flat histograms are rare in practical applications of histogram equalization. Thus, unlike its continuous counterpart, it cannot be proved (in general) that discrete histogram equalization results in a uniform histogram. However, as you will see shortly, using Eq. (3.3-8) has the general tendency to spread the histogram of the input image so that the intensity levels of the equalized image span a wider range of the intensity scale. The net result is contrast enhancement. ■

We discussed earlier in this section the many advantages of having intensity values that cover the entire gray scale. In addition to producing intensities 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 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 3.10) that this inverse transformation satisfies conditions (a′) and (b) only if none of the levels, r_k, $k = 0, 1, 2, \ldots, L - 1$, are missing from the input image, which in turn means that none of the components of the image histogram are zero. 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.

EXAMPLE 3.6: Histogram equalization.

■ The left column in Fig. 3.20 shows the four images from Fig. 3.16, and the center column shows the result of performing histogram equalization on each of these images. The first three results from top to bottom show significant improvement. As expected, histogram equalization did not have much effect on the fourth image because the intensities of this image already span the full intensity scale. Figure 3.21 shows the transformation functions used to generate the equalized images in Fig. 3.20. These functions were generated using Eq. (3.3-8). Observe that transformation (4) has a nearly linear shape, indicating that the inputs were mapped to nearly equal outputs.

The third column in Fig. 3.20 shows the histograms of the equalized images. 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 basic difference between the images on the left column is one of contrast, not content. In other words, because the images have the same content, the increase in contrast resulting from histogram equalization was enough to render any intensity differences in the equalized images visually indistinguishable. Given the significant contrast differences between the original images, this example illustrates the power of histogram equalization as an adaptive contrast 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*.

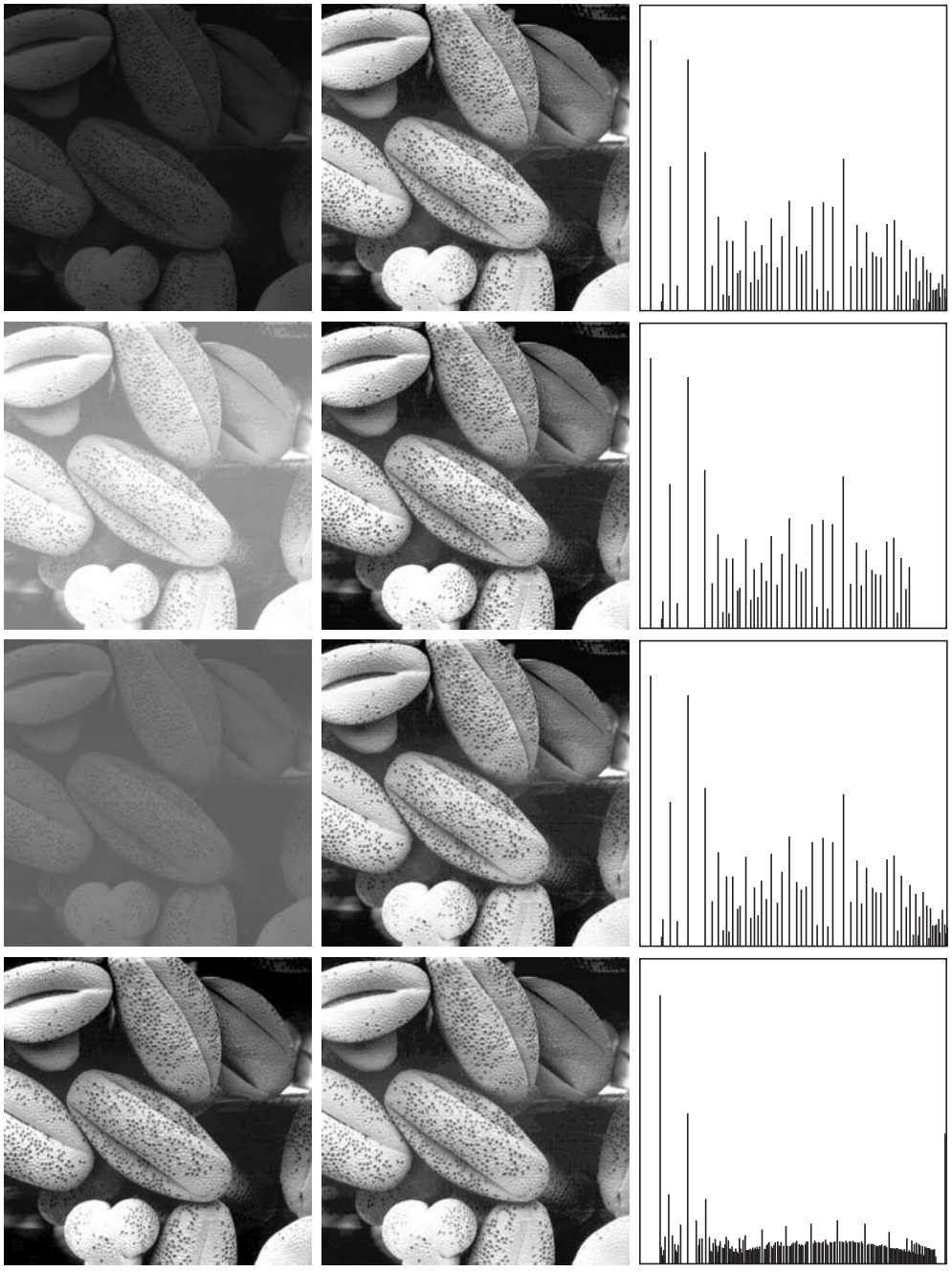


FIGURE 3.20 Left column: images from Fig. 3.16. Center column: corresponding histogram-equalized images. Right column: histograms of the images in the center column.

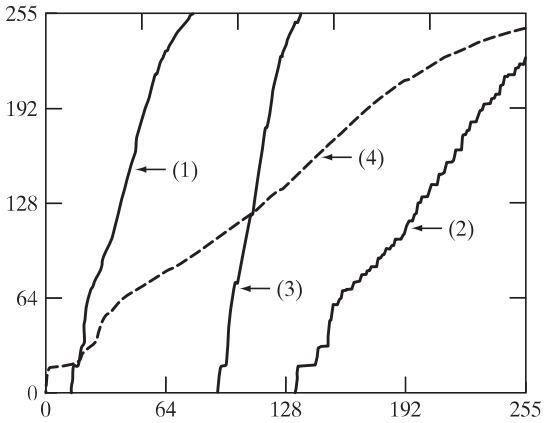


FIGURE 3.21 Transformation functions for histogram equalization. Transformations (1) through (4) were obtained from the histograms of the images (from top to bottom) in the left column of Fig. 3.20 using Eq. (3.3-8).

Let us return for a moment to continuous intensities 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 the intensity 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

$$s = T(r) = (L - 1)\int_0^r p_r(w)\,dw \tag{3.3-10}$$

where, as before, w is a dummy variable of integration. We recognize this expression as the continuous version of histogram equalization given in Eq. (3.3-4).

Suppose next that we define a random variable z with the property

$$G(z) = (L - 1)\int_0^z p_z(t)\,dt = s \tag{3.3-11}$$

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

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

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.

Equations (3.3-10) through (3.3-12) show that an image whose intensity levels have a specified probability density function can be obtained from a given image by using the following procedure:

1. Obtain $p_r(r)$ from the input image and use Eq. (3.3-10) to obtain the values of s.
2. Use the specified PDF in Eq. (3.3-11) to obtain the transformation function $G(z)$.

3. Obtain the inverse transformation $z = G^{-1}(s)$; because z is obtained from s, this process is a *mapping* from s to z, the latter being the desired values.
4. Obtain the output image by first equalizing the input image using Eq. (3.3-10); the pixel values in this image are the s values. For each pixel with value s in the equalized image, perform the inverse mapping $z = G^{-1}(s)$ to obtain the corresponding pixel in the output image. When all pixels have been thus processed, the PDF of the output image will be equal to the specified PDF.

EXAMPLE 3.7: Histogram specification.

■ Assuming continuous intensity values, suppose that an image has the intensity PDF $p_r(r) = 2r/(L-1)^2$ for $0 \le r \le (L-1)$ and $p_r(r) = 0$ for other values of r. Find the transformation function that will produce an image whose intensity PDF is $p_z(z) = 3z^2/(L-1)^3$ for $0 \le z \le (L-1)$ and $p_z(z) = 0$ for other values of z.

First, we find the histogram equalization transformation for the interval $[0, L-1]$:

$$s = T(r) = (L-1)\int_0^r p_r(w)\,dw = \frac{2}{(L-1)}\int_0^r w\,dw = \frac{r^2}{(L-1)}$$

By definition, this transformation is 0 for values outside the range $[0, L-1]$. Squaring the values of the input intensities and dividing them by $(L-1)^2$ will produce an image whose intensities, s, have a uniform PDF because this is a histogram-equalization transformation, as discussed earlier.

We are interested in an image with a specified histogram, so we find next

$$G(z) = (L-1)\int_0^z p_z(w)\,dw = \frac{3}{(L-1)^2}\int_0^z w^2\,dw = \frac{z^3}{(L-1)^2}$$

over the interval $[0, L-1]$; this function is 0 elsewhere by definition. Finally, we require that $G(z) = s$, but $G(z) = z^3/(L-1)^2$; so $z^3/(L-1)^2 = s$, and we have

$$z = \left[(L-1)^2 s\right]^{1/3}$$

So, if we multiply every histogram equalized pixel by $(L-1)^2$ and raise the product to the power 1/3, the result will be an image whose intensities, z, have the PDF $p_z(z) = 3z^2/(L-1)^3$ in the interval $[0, L-1]$, as desired.

Because $s = r^2/(L-1)$ we can generate the z's directly from the intensities, r, of the input image:

$$z = \left[(L-1)^2 s\right]^{1/3} = \left[(L-1)^2 \frac{r^2}{(L-1)}\right]^{1/3} = \left[(L-1)r^2\right]^{1/3}$$

Thus, squaring the value of each pixel in the original image, multiplying the result by $(L-1)$, and raising the product to the power 1/3 will yield an image

whose intensity levels, z, have the specified PDF. We see that the intermediate step of equalizing the input image can be skipped; all we need is to obtain the transformation function $T(r)$ that maps r to s. Then, the two steps can be combined into a single transformation from r to z. ■

As the preceding example shows, histogram specification is straightforward in principle. In practice, a common difficulty is finding meaningful analytical expressions for $T(r)$ and G^{-1}. Fortunately, the problem is simplified significantly when dealing with discrete quantities. The price paid is the same as for 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 the histogram equalization transformation in Eq. (3.3-8), which we repeat here for convenience:

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

where, as before, MN is the total number of pixels in the image, n_j is the number of pixels that have intensity value r_j, and L is the total number of possible intensity levels in the image. Similarly, given a specific value of s_k, the discrete formulation of Eq. (3.3-11) involves computing the transformation function

$$G(z_q) = (L-1)\sum_{i=0}^{q} p_z(z_i) \tag{3.3-14}$$

for a value of q, so that

$$G(z_q) = s_k \tag{3.3-15}$$

where $p_z(z_i)$, is the ith value of the specified histogram. As before, we find the desired value z_q by obtaining the inverse transformation:

$$z_q = G^{-1}(s_k) \tag{3.3-16}$$

In other words, this operation gives a value of z for each value of s; thus, it performs a *mapping* from s to z.

In practice, we do not need to compute the inverse of G. Because we deal with intensity levels that are integers (e.g., 0 to 255 for an 8-bit image), it is a simple matter to compute all the possible values of G using Eq. (3.3-14) for $q = 0, 1, 2, \ldots, L-1$. These values are scaled and rounded to their nearest integer values spanning the range $[0, L-1]$. The values are stored in a table. Then, given a particular value of s_k, we look for the closest match in the values stored in the table. If, for example, the 64th entry in the table is the closest to s_k, then $q = 63$ (recall that we start counting at 0) and z_{63} is the best solution to Eq. (3.3-15). Thus, the given value s_k would be associated with z_{63} (i.e., that

specific value of s_k would *map* to z_{63}). Because the zs are intensities used as the basis for specifying the histogram $p_z(z)$, it follows that $z_0 = 0$, $z_1 = 1, \ldots, z_{L-1} = L - 1$, so z_{63} would have the intensity value 63. By repeating this procedure, we would find the mapping of each value of s_k to the value of z_q that is the closest solution to Eq. (3.3-15). These mappings are the solution to the histogram-specification problem.

Recalling that the s_ks are the values of the histogram-equalized image, we may summarize the histogram-specification procedure as follows:

1. Compute the histogram $p_r(r)$ of the given image, and use it to find the histogram equalization transformation in Eq. (3.3-13). Round the resulting values, s_k, to the integer range $[0, L - 1]$.
2. Compute all values of the transformation function G using the Eq. (3.3-14) for $q = 0, 1, 2, \ldots, L - 1$, where $p_z(z_i)$ are the values of the specified histogram. Round the values of G to integers in the range $[0, L - 1]$. Store the values of G in a table.
3. For every value of s_k, $k = 0, 1, 2, \ldots, L - 1$, use the stored values of G from step 2 to find the corresponding value of z_q so that $G(z_q)$ is closest to s_k and store these mappings from s to z. When more than one value of z_q satisfies the given s_k (i.e., the mapping is not unique), choose the smallest value by convention.
4. Form the histogram-specified image by first histogram-equalizing the input image and then mapping every equalized pixel value, s_k, of this image to the corresponding value z_q in the histogram-specified image using the mappings found in step 3. As in the continuous case, the intermediate step of equalizing the input image is conceptual. It can be skipped by combining the two transformation functions, T and G^{-1}, as Example 3.8 shows.

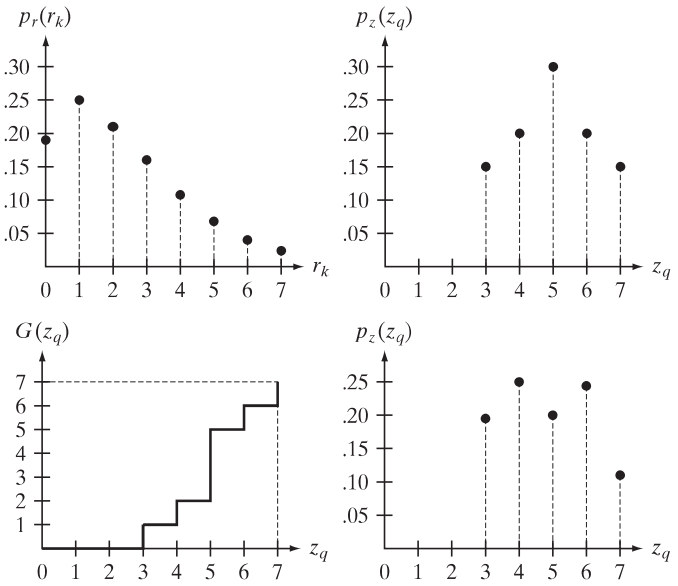
As mentioned earlier, for G^{-1} to satisfy conditions (a′) and (b), G has to be strictly monotonic, which, according to Eq. (3.3-14), means that none of the values $p_z(z_i)$ of the specified histogram can be zero (Problem 3.10). When working with discrete quantities, the fact that this condition may not be satisfied is not a serious implementation issue, as step 3 above indicates. The following example illustrates this numerically.

EXAMPLE 3.8: A simple example of histogram specification.

■ Consider again the 64×64 hypothetical image from Example 3.5, whose histogram is repeated in Fig. 3.22(a). It is desired to transform this histogram so that it will have the values specified in the second column of Table 3.2. Figure 3.22(b) shows a sketch of this histogram.

The first step in the procedure is to obtain the scaled histogram-equalized values, which we did in Example 3.5:

$$s_0 = 1 \quad s_2 = 5 \quad s_4 = 7 \quad s_6 = 7$$
$$s_1 = 3 \quad s_3 = 6 \quad s_5 = 7 \quad s_7 = 7$$



a b
c d

FIGURE 3.22
(a) Histogram of a 3-bit image. (b) Specified histogram. (c) Transformation function obtained from the specified histogram. (d) Result of performing histogram specification. Compare (b) and (d).

In the next step, we compute all the values of the transformation function, G, using Eq. (3.3-14):

$$G(z_0) = 7\sum_{j=0}^{0} p_z(z_j) = 0.00$$

Similarly,

$$G(z_1) = 7\sum_{j=0}^{1} p_z(z_j) = 7\big[p(z_0) + p(z_1)\big] = 0.00$$

and

$$G(z_2) = 0.00 \quad G(z_4) = 2.45 \quad G(z_6) = 5.95$$

$$G(z_3) = 1.05 \quad G(z_5) = 4.55 \quad G(z_7) = 7.00$$

TABLE 3.2
Specified and actual histograms (the values in the third column are from the computations performed in the body of Example 3.8).

z_q	Specified $p_z(z_q)$	Actual $p_z(z_k)$
$z_0 = 0$	0.00	0.00
$z_1 = 1$	0.00	0.00
$z_2 = 2$	0.00	0.00
$z_3 = 3$	0.15	0.19
$z_4 = 4$	0.20	0.25
$z_5 = 5$	0.30	0.21
$z_6 = 6$	0.20	0.24
$z_7 = 7$	0.15	0.11

As in Example 3.5, these fractional values are converted to integers in our valid range, [0, 7]. The results are:

$$G(z_0) = 0.00 \rightarrow 0 \qquad G(z_4) = 2.45 \rightarrow 2$$
$$G(z_1) = 0.00 \rightarrow 0 \qquad G(z_5) = 4.55 \rightarrow 5$$
$$G(z_2) = 0.00 \rightarrow 0 \qquad G(z_6) = 5.95 \rightarrow 6$$
$$G(z_3) = 1.05 \rightarrow 1 \qquad G(z_7) = 7.00 \rightarrow 7$$

These results are summarized in Table 3.3, and the transformation function is sketched in Fig. 3.22(c). Observe that G is not strictly monotonic, so condition (a′) is violated. Therefore, we make use of the approach outlined in step 3 of the algorithm to handle this situation.

In the third step of the procedure, we find the smallest value of z_q so that the value $G(z_q)$ is the closest to s_k. We do this for every value of s_k to create the required mappings from s to z. For example, $s_0 = 1$, and we see that $G(z_3) = 1$, which is a perfect match in this case, so we have the correspondence $s_0 \rightarrow z_3$. That is, every pixel whose value is 1 in the histogram equalized image would map to a pixel valued 3 (in the corresponding location) in the histogram-specified image. Continuing in this manner, we arrive at the mappings in Table 3.4.

In the final step of the procedure, we use the mappings in Table 3.4 to map every pixel in the histogram equalized image into a corresponding pixel in the newly created histogram-specified image. The values of the resulting histogram are listed in the third column of Table 3.2, and the histogram is sketched in Fig. 3.22(d). The values of $p_z(z_q)$ were obtained using the same procedure as in Example 3.5. For instance, we see in Table 3.4 that $s = 1$ maps to $z = 3$, and there are 790 pixels in the histogram-equalized image with a value of 1. Therefore, $p_z(z_3) = 790/4096 = 0.19$.

Although the final result shown in Fig. 3.22(d) does not match the specified histogram exactly, the general trend of moving the intensities toward the high end of the intensity scale definitely was achieved. As mentioned earlier, obtaining the histogram-equalized image as an intermediate step is useful for explaining the procedure, but this is not necessary. Instead, we could list the mappings from the rs to the ss and from the ss to the zs in a three-column

z_q	$G(z_q)$
$z_0 = 0$	0
$z_1 = 1$	0
$z_2 = 2$	0
$z_3 = 3$	1
$z_4 = 4$	2
$z_5 = 5$	5
$z_6 = 6$	6
$z_7 = 7$	7

TABLE 3.3
All possible values of the transformation function G scaled, rounded, and ordered with respect to z.

TABLE 3.4
Mappings of all the values of s_k into corresponding values of z_q.

s_k	→	z_q
1	→	3
3	→	4
5	→	5
6	→	6
7	→	7

table. Then, we would use those mappings to map the original pixels directly into the pixels of the histogram-specified image. ■

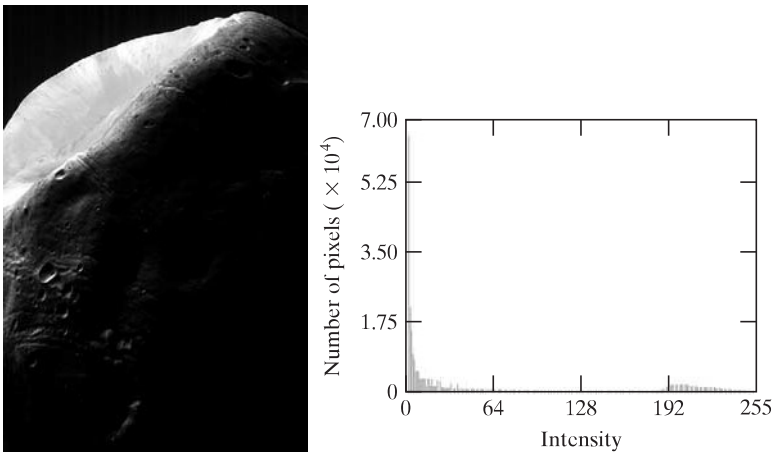
EXAMPLE 3.9: Comparison between histogram equalization and histogram matching.

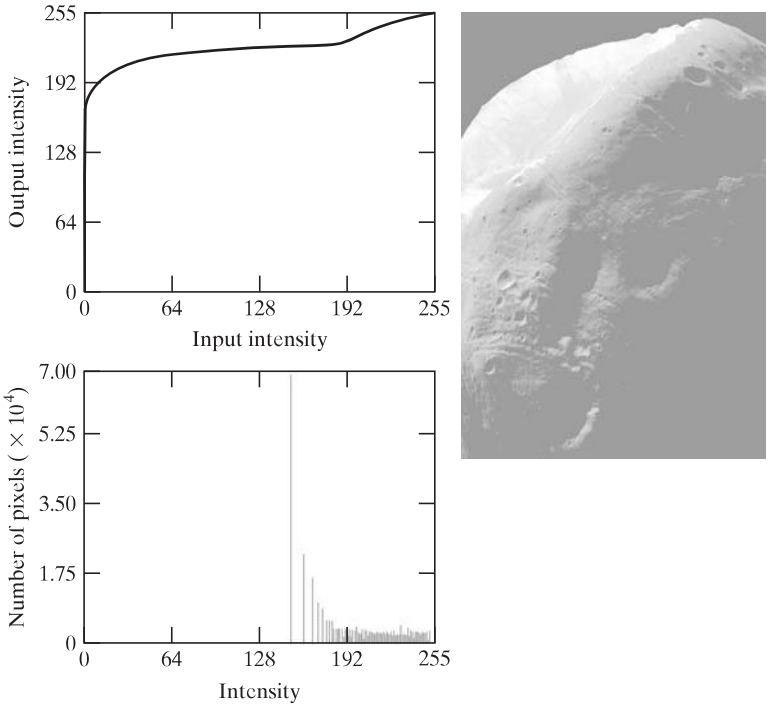
■ Figure 3.23(a) shows an image of the Mars moon, Phobos, taken by NASA's *Mars Global Surveyor*. Figure 3.23(b) shows the histogram of Fig. 3.23(a). The image is dominated by large, dark areas, resulting in a histogram characterized by a large concentration of pixels in the dark end of the gray scale. 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 3.24(a) shows the histogram equalization transformation [Eq. (3.3-8) or (3.3-13)] obtained from the histogram in Fig. 3.23(b). The most relevant characteristic of this transformation function is how fast it rises from intensity level 0 to a level near 190. This is caused by the large concentration of pixels in the input histogram having levels 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 image with a light, washed-out appearance. As Fig. 3.24(b) shows, this is indeed the

a b

FIGURE 3.23
(a) Image of the Mars moon Phobos taken by NASA's *Mars Global Surveyor*. (b) Histogram. (Original image courtesy of NASA.)





a b
c

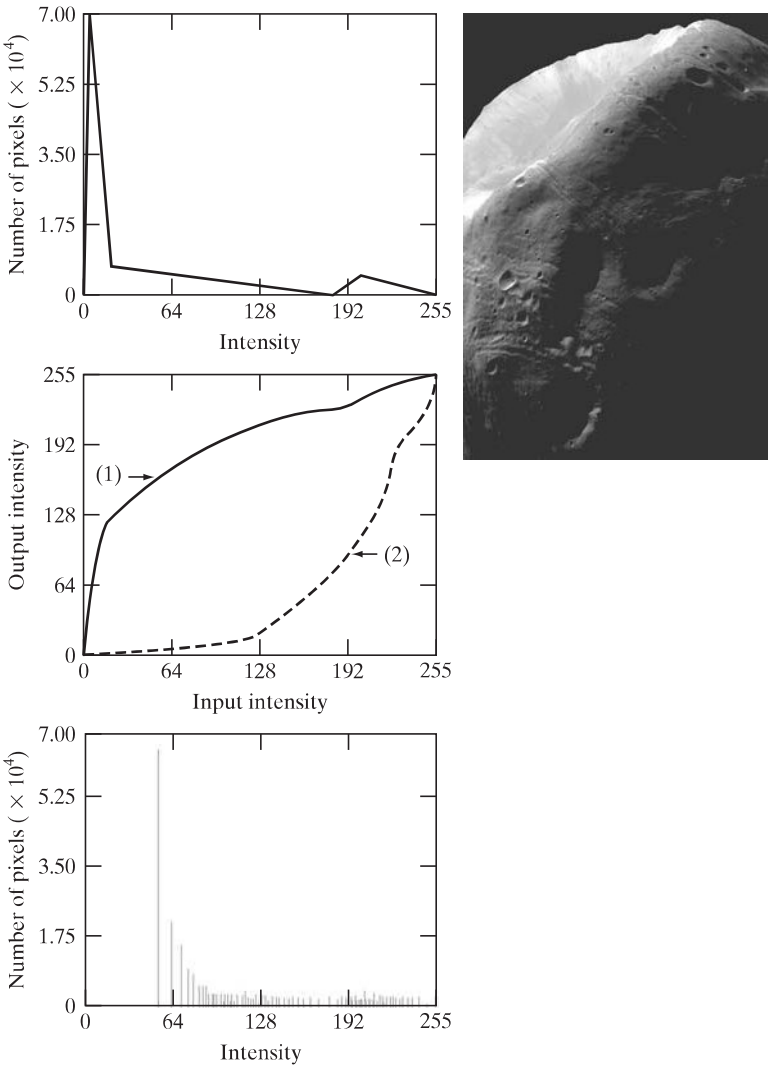
FIGURE 3.24
(a) Transformation function for histogram equalization. (b) Histogram-equalized image (note the washed-out appearance). (c) Histogram of (b).

case. The histogram of this image is shown in Fig. 3.24(c). Note how all the intensity levels are biased toward the upper one-half of the gray scale.

Because the problem with the transformation function in Fig. 3.24(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 3.25(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. 3.25(b). Similarly, the inverse transformation $G^{-1}(s)$ from Eq. (3.3-16) (obtained using the step-by-step procedure discussed earlier) is labeled transformation (2) in Fig. 3.25(b). The enhanced image in Fig. 3.25(c) was obtained by applying transformation (2) to the pixels of the histogram-equalized image in Fig. 3.24(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 appearance. Figure 3.25(d) shows the histogram of Fig. 3.25(c). The most distinguishing feature of this histogram is how its low end has shifted right toward the lighter region of the gray scale (but not excessively so), as desired. ■

a c
b
d

FIGURE 3.25
(a) Specified histogram.
(b) Transformations.
(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.

3.3.3 Local Histogram Processing

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 intensity distribution 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 intensity distribution in a neighborhood of every pixel in the image.

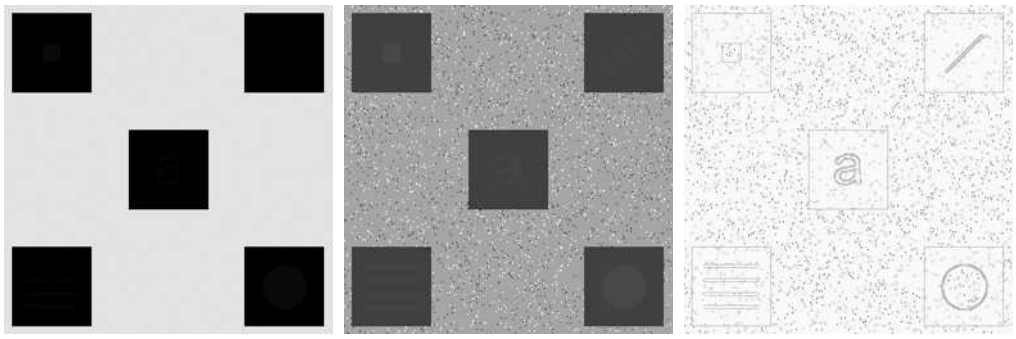
The histogram processing techniques previously described are easily adapted to local enhancement. The procedure is to define a neighborhood and move its center from pixel to pixel. At each location, the histogram of the points in the neighborhood is computed and either a histogram equalization or histogram specification transformation function is obtained. This function is then used to map the intensity 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. Because only one row or column of the neighborhood changes during a pixel-to-pixel translation of the neighborhood, updating the histogram obtained in the previous location with the new data introduced at each motion step is possible (Problem 3.12). This approach has obvious advantages over repeatedly computing the histogram of all pixels in the neighborhood region each time the region is moved one pixel location. Another approach used sometimes to reduce computation is to utilize nonoverlapping regions, but this method usually produces an undesirable "blocky" effect.

EXAMPLE 3.10: Local histogram equalization.

■ Figure 3.26(a) shows an 8-bit, 512×512 image that at first glance appears to contain five black squares on a gray background. The image is slightly noisy, but the noise is imperceptible. Figure 3.26(b) shows the result of global histogram equalization. As often is the case with histogram equalization of smooth, noisy regions, this image shows significant enhancement of the noise. Aside from the noise, however, Fig. 3.26(b) does not reveal any new significant details from the original, other than a very faint hint that the top left and bottom right squares contain an object. Figure 3.26(c) was obtained using local histogram equalization with a neighborhood of size 3×3. Here, we see significant detail contained within the dark squares. The intensity values of these objects were too close to the intensity of the large squares, and their sizes were too small, to influence global histogram equalization significantly enough to show this detail. ■

3.3.4 Using Histogram Statistics for Image Enhancement

Statistics obtained directly from an image histogram can be used for image enhancement. Let r denote a discrete random variable representing intensity values in the range $[0, L-1]$, and let $p(r_i)$ denote the normalized histogram



a b c

FIGURE 3.26 (a) Original image. (b) Result of global histogram equalization. (c) Result of local histogram equalization applied to (a), using a neighborhood of size 3 × 3.

component corresponding to value r_i. As indicated previously, we may view $p(r_i)$ as an estimate of the probability that intensity r_i occurs in the image from which the histogram was obtained.

As we discussed in Section 2.6.8, the nth moment of r about its mean is defined as

$$\mu_n(r) = \sum_{i=0}^{L-1} (r_i - m)^n p(r_i) \tag{3.3-17}$$

We follow convention in using m for the mean value. Do not confuse it with the same symbol used to denote the number of rows in an $m \times n$ neighborhood, in which we also follow notational convention.

where m is the mean (average intensity) value of r (i.e., the average intensity of the pixels in the image):

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

The second moment is particularly important:

$$\mu_2(r) = \sum_{i=0}^{L-1} (r_i - m)^2 p(r_i) \tag{3.3-19}$$

We recognize this expression as the intensity variance, normally denoted by σ^2 (recall that the standard deviation is the square root of the variance). Whereas the mean is a measure of average intensity, the variance (or standard deviation) is a measure of contrast in an image. Observe that all moments are computed easily using the preceding expressions once the histogram has been obtained from a given image.

When working with only the mean and variance, it is common practice to estimate them directly from the sample values, without computing the histogram. Appropriately, these estimates are called the *sample mean* and *sample variance*. They are given by the following familiar expressions from basic statistics:

$$m = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) \tag{3.3-20}$$

and

$$\sigma^2 = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[f(x, y) - m\right]^2 \tag{3.3-21}$$

for $x = 0, 1, 2, \ldots, M - 1$ and $y = 0, 1, 2, \ldots, N - 1$. In other words, as we know, the mean intensity of an image can be obtained simply by summing the values of all its pixels and dividing the sum by the total number of pixels in the image. A similar interpretation applies to Eq. (3.3-21). As we illustrate in the following example, the results obtained using these two equations are identical to the results obtained using Eqs. (3.3-18) and (3.3-19), provided that the histogram used in these equations is computed from the same image used in Eqs. (3.3-20) and (3.3-21).

The denominator in Eq. (3.3-21) is written sometimes as $MN - 1$ instead of MN. This is done to obtain a so-called *unbiased* estimate of the variance. However, we are more interested in Eqs. (3.3-21) and (3.3-19) agreeing when the histogram in the latter equation is computed from the same image used in Eq. (3.3-21). For this we require the MN term. The difference is negligible for any image of practical size.

EXAMPLE 3.11: Computing histogram statistics.

■ Before proceeding, it will be useful to work through a simple numerical example to fix ideas. Consider the following 2-bit image of size 5×5:

$$\begin{array}{ccccc} 0 & 0 & 1 & 1 & 2 \\ 1 & 2 & 3 & 0 & 1 \\ 3 & 3 & 2 & 2 & 0 \\ 2 & 3 & 1 & 0 & 0 \\ 1 & 1 & 3 & 2 & 2 \end{array}$$

The pixels are represented by 2 bits; therefore, $L = 4$ and the intensity levels are in the range $[0, 3]$. The total number of pixels is 25, so the histogram has the components

$$p(r_0) = \frac{6}{25} = 0.24; \quad p(r_1) = \frac{7}{25} = 0.28;$$

$$p(r_2) = \frac{7}{25} = 0.28; \quad p(r_3) = \frac{5}{25} = 0.20$$

where the numerator in $p(r_i)$ is the number of pixels in the image with intensity level r_i. We can compute the average value of the intensities in the image using Eq. (3.3-18):

$$\begin{aligned} m &= \sum_{i=0}^{3} r_i\, p(r_i) \\ &= (0)(0.24) + (1)(0.28) + (2)(0.28) + (3)(0.20) \\ &= 1.44 \end{aligned}$$

Letting $f(x, y)$ denote the preceding 5×5 array and using Eq. (3.3-20), we obtain

$$\begin{aligned} m &= \frac{1}{25}\sum_{x=0}^{4}\sum_{y=0}^{4} f(x, y) \\ &= 1.44 \end{aligned}$$

As expected, the results agree. Similarly, the result for the variance is the same (1.1264) using either Eq. (3.3-19) or (3.3-21). ■

We consider two uses of the mean and variance for enhancement purposes. The *global* mean and variance are computed over an entire image and are useful for gross adjustments in overall intensity and contrast. A more powerful use of these parameters 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 neighborhood about each pixel in an image.

Let (x, y) denote the coordinates of any pixel in a given image, and let S_{xy} denote a neighborhood (subimage) of specified size, centered on (x, y). The mean value of the pixels in this neighborhood is given by the expression

$$m_{S_{xy}} = \sum_{i=0}^{L-1} r_i p_{S_{xy}}(r_i) \tag{3.3-22}$$

where $p_{S_{xy}}$ is the histogram of the pixels in region S_{xy}. This histogram has L components, corresponding to the L possible intensity values in the input image. However, many of the components are 0, depending on the size of S_{xy}. For example, if the neighborhood is of size 3×3 and $L = 256$, only between 1 and 9 of the 256 components of the histogram of the neighborhood will be nonzero. These non-zero values will correspond to the number of different intensities in S_{xy} (the maximum number of possible different intensities in a 3×3 region is 9, and the minimum is 1).

The variance of the pixels in the neighborhood similarly is given by

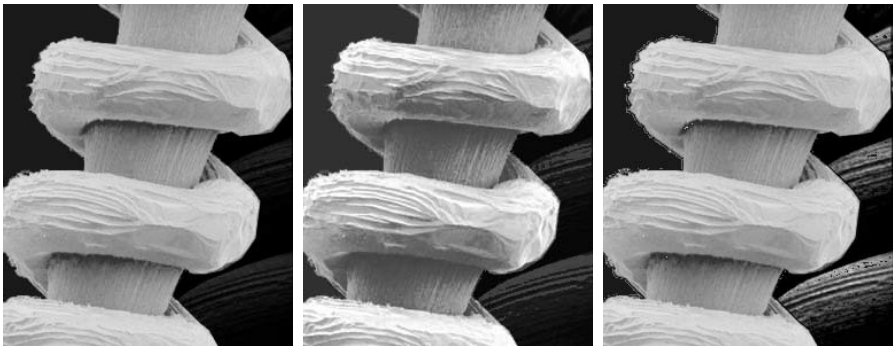
$$\sigma^2_{S_{xy}} = \sum_{i=0}^{L-1} (r_i - m_{S_{xy}})^2 p_{S_{xy}}(r_i) \tag{3.3-23}$$

As before, the local mean is a measure of average intensity in neighborhood S_{xy}, and the local variance (or standard deviation) is a measure of intensity contrast in that neighborhood. Expressions analogous to (3.3-20) and (3.3-21) can be written for neighborhoods. We simply use the pixel values in the neighborhoods in the summations and the number of pixels in the neighborhood in the denominator.

As the following example illustrates, 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.

EXAMPLE 3.12: Local enhancement using histogram statistics.

■ Figure 3.27(a) 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, dark side of the image, but it is almost imperceptible, and its size and other characteristics certainly are not easily discernable. Local enhancement by contrast manipulation is an ideal approach to problems such as this, in which parts of an image may contain hidden features.



a b c

FIGURE 3.27 (a) SEM image of a tungsten filament magnified approximately 130×. (b) Result of global histogram equalization. (c) Image enhanced using local histogram statistics. (Original image courtesy of Mr. Michael Shaffer, Department of Geological Sciences, University of Oregon, Eugene.)

In this particular case, the problem is to enhance dark areas while leaving the light area as unchanged as possible because it does not 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 average local intensity, $m_{S_{xy}}$, to the average image intensity, called the *global mean* and denoted m_G. This quantity is obtained with Eq. (3.3-18) or (3.3-20) using 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.

Because 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. We consider the pixel at a point (x, y) as a candidate for enhancement if $\sigma_{S_{xy}} \leq k_2 \sigma_G$, where σ_G is the *global standard deviation* obtained using Eqs. (3.3-19) or (3.3-21) 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 constant areas, whose standard deviation is zero. Thus, we also set a lower limit on the local standard deviation by requiring that $k_1 \sigma_G \leq \sigma_{S_{xy}}$, 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 intensity level relative to the rest of the image. Pixels that do not meet the enhancement conditions are not changed.

We summarize the preceding approach as follows. Let $f(x, y)$ represent the value of an image at any image coordinates (x, y), and let $g(x, y)$ represent the corresponding enhanced value at those coordinates. Then,

$$g(x, y) = \begin{cases} E \cdot f(x, y) & \text{if } m_{S_{xy}} \leq k_0 m_G \text{ AND } k_1 \sigma_G \leq \sigma_{S_{xy}} \leq k_2 \sigma_G \\ f(x, y) & \text{otherwise} \end{cases} \tag{3.3-24}$$

for $x = 0, 1, 2, \ldots, M - 1$ and $y = 0, 1, 2, \ldots, N - 1$, where, as indicated above, E, k_0, k_1, and k_2 are specified parameters, m_G is the global mean of the input image, and σ_G is its standard deviation. Parameters $m_{S_{xy}}$ and $\sigma_{S_{xy}}$ are the local mean and standard deviation, respectively. As usual, M and N are the row and column image dimensions.

Choosing the parameters in Eq. (3.3-24) generally 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 less than half the global mean because we can see 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 difficult in general, but their choice definitely must be guided by a logical analysis of the enhancement problem at hand. Finally, the size of the local area S_{xy} should be as small as possible in order to preserve detail and keep the computational burden as low as possible. We chose a region of size 3×3.

As a basis for comparison, we enhanced the image using global histogram equalization. Figure 3.27(b) shows the result. The dark area was improved but details still are difficult to discern, and the light areas were changed, something we did not want to do. Figure 3.27(c) shows the result of using the local statistics method explained above. In comparing this image with the original in Fig. 3.27(a) or the histogram equalized result in Fig. 3.27(b), we note the obvious detail that has been brought out on the right side of Fig. 3.27(c). Observe, for example, the clarity of the ridges in the dark filaments. It is noteworthy that the light-intensity areas on the left were left nearly intact, which was one of our initial objectives. ■

3.4 Fundamentals of Spatial Filtering

In this section, we introduce several basic concepts underlying the use of spatial filters for image processing. Spatial filtering is one of the principal tools used in this field for a broad spectrum of applications, so it is highly advisable that you develop a solid understanding of these concepts. As mentioned at the beginning of this chapter, the examples in this section deal mostly with the use of spatial filters for image enhancement. Other applications of spatial filtering are discussed in later chapters.

The name *filter* is borrowed from frequency domain processing, which is the topic of the next chapter, where "filtering" refers to accepting (passing) or rejecting certain frequency components. For example, a filter that passes low frequencies is called a *lowpass* filter. The net effect produced by a lowpass filter is to blur (smooth) an image. We can accomplish a similar smoothing directly on the image itself by using spatial filters (also called spatial *masks*, *kernels*, *templates*, and *windows*). In fact, as we show in Chapter 4, there is a one-to-one correspondence between linear spatial filters and filters in the frequency domain. However, spatial filters offer considerably more versatility because, as you will see later, they can be used also for nonlinear filtering, something we cannot do in the frequency domain.

See Section 2.6.2 regarding linearity.

3.4.1 The Mechanics of Spatial Filtering

In Fig. 3.1, we explained briefly that a spatial filter consists of (1) a *neighborhood*, (typically a small rectangle), and (2) a *predefined operation* that is performed on the image pixels encompassed by the neighborhood. Filtering creates a new pixel with coordinates equal to the coordinates of the center of the neighborhood, and whose value is the result of the filtering operation.[†] A processed (filtered) image is generated as the center of the filter visits each pixel in the input image. If the operation performed on the image pixels is linear, then the filter is called a *linear spatial filter*. Otherwise, the filter is *nonlinear*. We focus attention first on linear filters and then illustrate some simple nonlinear filters. Section 5.3 contains a more comprehensive list of nonlinear filters and their application.

Figure 3.28 illustrates the mechanics of linear spatial filtering using a 3×3 neighborhood. At any point (x, y) in the image, the response, $g(x, y)$, of the filter is the sum of products of the filter coefficients and the image pixels encompassed by the filter:

$$g(x, y) = w(-1, -1)f(x - 1, y - 1) + w(-1, 0)f(x - 1, y) + \ldots + w(0, 0)f(x, y) + \ldots + w(1, 1)f(x + 1, y + 1)$$

Observe that the center coefficient of the filter, $w(0, 0)$, aligns with the pixel at location (x, y). For a mask of size $m \times n$, we assume that $m = 2a + 1$ and $n = 2b + 1$, where a and b are positive integers. This means that our focus in the following discussion is on filters of odd size, with the smallest being of size 3×3. In general, linear spatial filtering of an image of size $M \times N$ with a filter of size $m \times n$ is given by the expression:

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

where x and y are varied so that each pixel in w visits every pixel in f.

It certainly is possible to work with filters of even size or mixed even and odd sizes. However, working with odd sizes simplifies indexing and also is more intuitive because the filters have centers falling on integer values.

[†] The filtered pixel value typically is assigned to a corresponding location in a new image created to hold the results of filtering. It is seldom the case that filtered pixels replace the values of the corresponding location in the original image, as this would change the content of the image while filtering still is being performed.

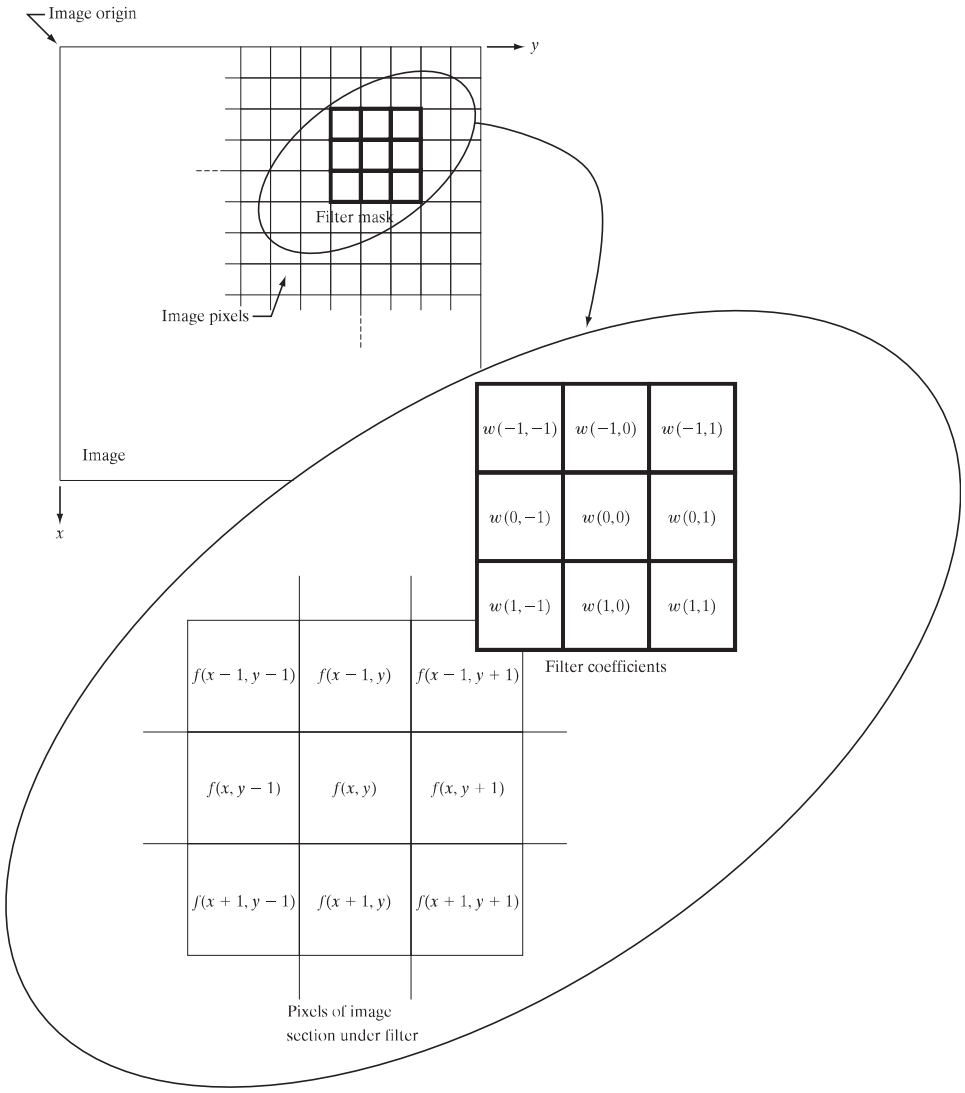


FIGURE 3.28 The mechanics of linear spatial filtering using a 3×3 filter mask. The form chosen to denote the coordinates of the filter mask coefficients simplifies writing expressions for linear filtering.

3.4.2 Spatial Correlation and Convolution

There are two closely related concepts that must be understood clearly when performing linear spatial filtering. One is *correlation* and the other is *convolution*. Correlation is the process of moving a filter mask over the image and computing the sum of products at each location, exactly as explained in the previous section. The mechanics of convolution are the same, except that the filter is first rotated by 180°. The best way to explain the differences between the two concepts is by example. We begin with a 1-D illustration.

Figure 3.29(a) shows a 1-D function, f, and a filter, w, and Fig. 3.29(b) shows the starting position to perform correlation. The first thing we note is that there

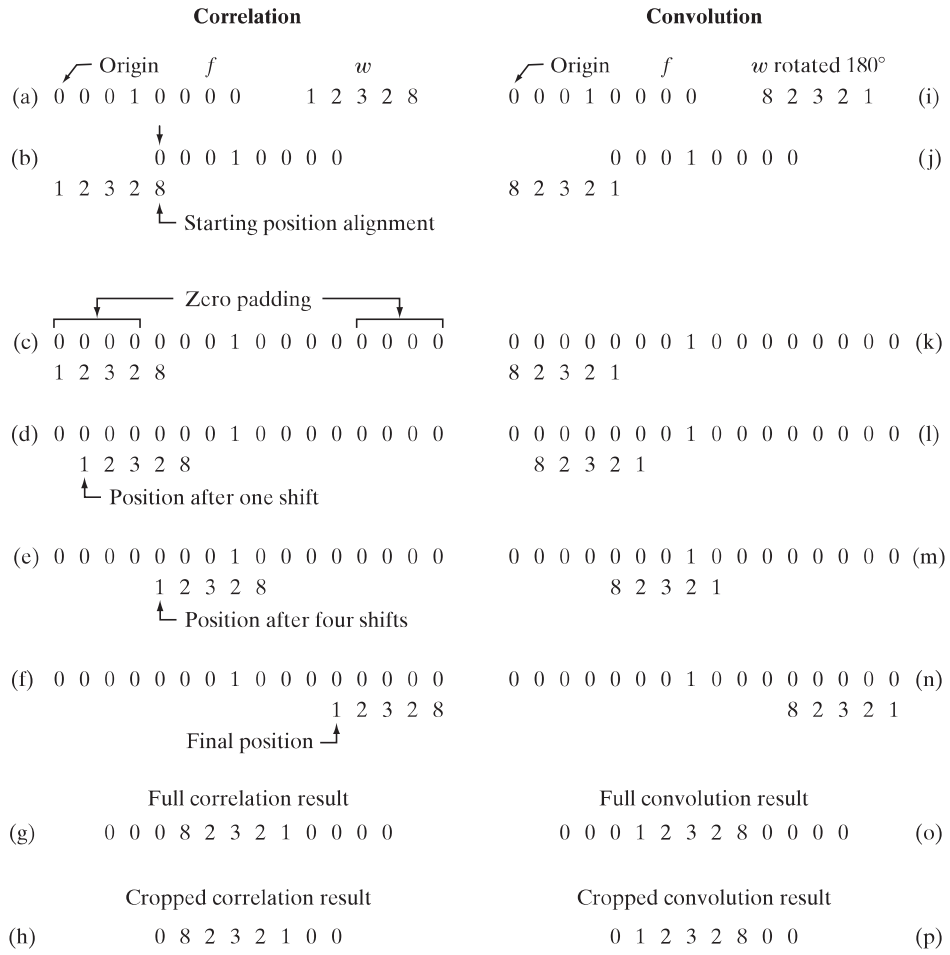


FIGURE 3.29 Illustration of 1-D correlation and convolution of a filter with a discrete unit impulse. Note that correlation and convolution are functions of *displacement*.

are parts of the functions that do not overlap. The solution to this problem is to pad f with enough 0s on either side to allow each pixel in w to visit every pixel in f. If the filter is of size m, we need $m - 1$ 0s on either side of f. Figure 3.29(c) shows a properly padded function. The first value of correlation is the sum of products of f and w for the initial position shown in Fig. 3.29(c) (the sum of products is 0). This corresponds to a displacement $x = 0$. To obtain the second value of correlation, we shift w one pixel location to the right (a displacement of $x = 1$) and compute the sum of products. The result again is 0. In fact, the first nonzero result is when $x = 3$, in which case the 8 in w overlaps the 1 in f and the result of correlation is 8. Proceeding in this manner, we obtain the full correlation result in Fig. 3.29(g). Note that it took 12 values of x (i.e., $x = 0, 1, 2, \ldots, 11$) to fully slide w past f so that each pixel in w visited every pixel in f. Often, we like to work with correlation arrays that are the same size as f, in which case we crop the full correlation to the size of the original function, as Fig. 3.29(h) shows.

Zero padding is not the only option. For example, we could duplicate the value of the first and last element $m - 1$ times on each side of f, or mirror the first and last $m - 1$ elements and use the mirrored values for padding.

There are two important points to note from the discussion in the preceding paragraph. First, correlation is a function of *displacement* of the filter. In other words, the first value of correlation corresponds to zero displacement of the filter, the second corresponds to one unit displacement, and so on. The second thing to notice is that correlating a filter w with a function that contains all 0s and a single 1 yields a result that is a *copy* of w, but *rotated* by 180°. We call a function that contains a single 1 with the rest being 0s a *discrete unit impulse*. So we conclude that correlation of a function with a discrete unit impulse yields a rotated version of the function at the location of the impulse.

The concept of convolution is a cornerstone of linear system theory. As you will learn in Chapter 4, a fundamental property of convolution is that convolving a function with a unit impulse yields a copy of the function at the location of the impulse. We saw in the previous paragraph that correlation yields a copy of the function also, but rotated by 180°. Therefore, if we *pre-rotate* the filter and perform the same sliding sum of products operation, we should be able to obtain the desired result. As the right column in Fig. 3.29 shows, this indeed is the case. Thus, we see that to perform convolution all we do is rotate one function by 180° and perform the same operations as in correlation. As it turns out, it makes no difference which of the two functions we rotate.

Note that rotation by 180° is equivalent to flipping the function horizontally.

The preceding concepts extend easily to images, as Fig. 3.30 shows. For a filter of size $m \times n$, we pad the image with a minimum of $m - 1$ rows of 0s at the top and bottom and $n - 1$ columns of 0s on the left and right. In this case, m and n are equal to 3, so we pad f with two rows of 0s above and below and two columns of 0s to the left and right, as Fig. 3.30(b) shows. Figure 3.30(c) shows the initial position of the filter mask for performing correlation, and Fig. 3.30(d) shows the full correlation result. Figure 3.30(e) shows the corresponding cropped result. Note again that the result is rotated by 180°. For convolution, we pre-rotate the mask as before and repeat the sliding sum of products just explained. Figures 3.30(f) through (h) show the result. You see again that convolution of a function with an impulse copies the function at the location of the impulse. It should be clear that, if the filter mask is symmetric, correlation and convolution yield the same result.

In 2-D, rotation by 180° is equivalent to flipping the mask along one axis and then the other.

If, instead of containing a single 1, image f in Fig. 3.30 had contained a region identically equal to w, the value of the correlation function (after normalization) would have been maximum when w was centered on that region of f. Thus, as you will see in Chapter 12, correlation can be used also to find *matches* between images.

Summarizing the preceding discussion in equation form, we have that the correlation of a filter $w(x, y)$ of size $m \times n$ with an image $f(x, y)$, denoted as $w(x, y) \star f(x, y)$, is given by the equation listed at the end of the last section, which we repeat here for convenience:

$$w(x, y) \star f(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s, t) f(x + s, y + t) \tag{3.4-1}$$

This equation is evaluated for all values of the displacement variables x and y so that all elements of w visit every pixel in f, where we assume that f has been padded appropriately. As explained earlier, $a = (m - 1)/2$, $b = (n - 1)/2$, and we assume for notational convenience that m and n are odd integers.

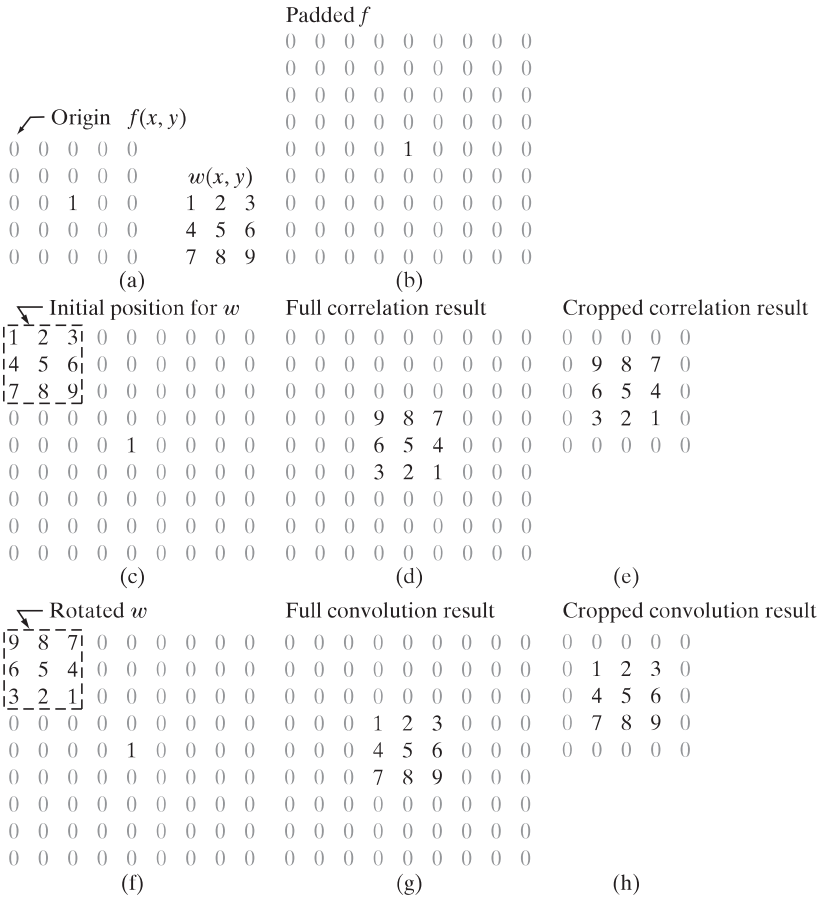


FIGURE 3.30 Correlation (middle row) and convolution (last row) of a 2-D filter with a 2-D discrete, unit impulse. The 0s are shown in gray to simplify visual analysis.

In a similar manner, the convolution of $w(x, y)$ and $f(x, y)$, denoted by $w(x, y) \star f(x, y)$,[†] is given by the expression

$$w(x, y) \star f(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s, t) f(x - s, y - t) \qquad (3.4\text{-}2)$$

where the minus signs on the right flip f (i.e., rotate it by 180°). Flipping and shifting f instead of w is done for notational simplicity and also to follow convention. The result is the same. As with correlation, this equation is evaluated for all values of the displacement variables x and y so that every element of w visits every pixel in f, which we assume has been padded appropriately. You should expand Eq. (3.4-2) for a 3×3 mask and convince yourself that the result using this equation is identical to the example in Fig. 3.30. In practice, we frequently work with an algorithm that implements

Often, when the meaning is clear, we denote the result of correlation or convolution by a function $g(x, y)$, instead of writing $w(x, y) \star f(x, y)$ or $w(x, y) \star f(x, y)$. For example, see the equation at the end of the previous section, and Eq. (3.5-1).

[†] Because correlation and convolution are commutative, we have that $w(x, y) \star f(x, y) = f(x, y) \star w(x, y)$ and $w(x, y) \star f(x, y) = f(x, y) \star w(x, y)$.

Eq. (3.4-1). If we want to perform correlation, we input w into the algorithm; for convolution, we input w rotated by 180°. The reverse is true if an algorithm that implements Eq. (3.4-2) is available instead.

As mentioned earlier, convolution is a cornerstone of linear system theory. As you will learn in Chapter 4, the property that the convolution of a function with a unit impulse copies the function at the location of the impulse plays a central role in a number of important derivations. We will revisit convolution in Chapter 4 in the context of the Fourier transform and the convolution theorem. Unlike Eq. (3.4-2), however, we will be dealing with convolution of functions that are of the same size. The form of the equation is the same, but the limits of summation are different.

Using correlation or convolution to perform spatial filtering is a matter of preference. In fact, because either Eq. (3.4-1) or (3.4-2) can be made to perform the function of the other by a simple rotation of the filter, what is important is that the filter mask used in a given filtering task be specified in a way that corresponds to the intended operation. All the linear spatial filtering results in this chapter are based on Eq. (3.4-1).

Finally, we point out that you are likely to encounter the terms, *convolution filter*, *convolution mask* or *convolution kernel* in the image processing literature. As a rule, these terms are used to denote a spatial filter, and not necessarily that the filter will be used for true convolution. Similarly, "convolving a mask with an image" often is used to denote the sliding, sum-of-products process we just explained, and does not necessarily differentiate between correlation and convolution. Rather, it is used generically to denote either of the two operations. This imprecise terminology is a frequent source of confusion.

3.4.3 Vector Representation of Linear Filtering

When interest lies in the characteristic response, R, of a mask either for correlation or convolution, it is convenient sometimes to write the sum of products as

$$\begin{aligned} R &= w_1 z_1 + w_2 z_2 + \ldots + w_{mn} z_{mn} \\ &= \sum_{k=1}^{mn} w_k z_k \\ &= \mathbf{w}^T \mathbf{z} \end{aligned} \tag{3.4-3}$$

Consult the Tutorials section of the book Web site for a brief review of vectors and matrices.

where the ws are the coefficients of an $m \times n$ filter and the zs are the corresponding image intensities encompassed by the filter. If we are interested in using Eq. (3.4-3) for correlation, we use the mask as given. To use the same equation for convolution, we simply rotate the mask by 180°, as explained in the last section. It is implied that Eq. (3.4-3) holds for a particular pair of coordinates (x, y). You will see in the next section why this notation is convenient for explaining the characteristics of a given linear filter.

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

FIGURE 3.31 Another representation of a general 3×3 filter mask.

As an example, Fig. 3.31 shows a general 3×3 mask with coefficients labeled as above. In this case, Eq. (3.4-3) becomes

$$\begin{aligned} R &= w_1 z_1 + w_2 z_2 + \ldots + w_9 z_9 \\ &= \sum_{k=1}^{9} w_k z_k \\ &= \mathbf{w}^T \mathbf{z} \end{aligned} \tag{3.4-4}$$

where $\mathbf{w}$ and $\mathbf{z}$ are 9-dimensional vectors formed from the coefficients of the mask and the image intensities encompassed by the mask, respectively.

3.4.4 Generating Spatial Filter Masks

Generating an $m \times n$ *linear* spatial filter requires that we specify mn mask coefficients. In turn, these coefficients are selected based on what the filter is supposed to do, keeping in mind that all we can do with linear filtering is to implement a sum of products. For example, suppose that we want to replace the pixels in an image by the average intensity of a 3×3 neighborhood centered on those pixels. The average value at any location (x, y) in the image is the sum of the nine intensity values in the 3×3 neighborhood centered on (x, y) divided by 9. Letting z_i, $i = 1, 2, \ldots, 9$, denote these intensities, the average is

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

But this is the same as Eq. (3.4-4) with coefficient values $w_i = 1/9$. In other words, a linear filtering operation with a 3×3 mask whose coefficients are $1/9$ implements the desired averaging. As we discuss in the next section, this operation results in image smoothing. We discuss in the following sections a number of other filter masks based on this basic approach.

In some applications, we have a continuous function of two variables, and the objective is to obtain a spatial filter mask based on that function. For example, a Gaussian function of two variables has the basic form

$$h(x, y) = e^{-\frac{x^2 + y^2}{2\sigma^2}}$$

where σ is the standard deviation and, as usual, we assume that coordinates x and y are integers. To generate, say, a 3×3 filter mask from this function, we

sample it about its center. Thus, $w_1 = h(-1, -1), w_2 = h(-1, 0), \ldots, w_9 = h(1, 1)$. An $m \times n$ filter mask is generated in a similar manner. Recall that a 2-D Gaussian function has a bell shape, and that the standard deviation controls the "tightness" of the bell.

Generating a *nonlinear* filter requires that we specify the size of a neighborhood and the operation(s) to be performed on the image pixels contained in the neighborhood. For example, recalling that the max operation is nonlinear (see Section 2.6.2), a 5×5 max filter centered at an arbitrary point (x, y) of an image obtains the maximum intensity value of the 25 pixels and assigns that value to location (x, y) in the processed image. Nonlinear filters are quite powerful, and in some applications can perform functions that are beyond the capabilities of linear filters, as we show later in this chapter and in Chapter 5.

3.5 Smoothing Spatial Filters

Smoothing filters are used for blurring and for noise reduction. Blurring is used in preprocessing tasks, 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.

3.5.1 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*. As mentioned in the previous section, 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 intensity levels in the neighborhood defined by the filter mask, this process results in an image with reduced "sharp" transitions in intensities. Because random noise typically consists of sharp transitions in intensity 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 intensity transitions, 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 from using an insufficient number of intensity 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 3.32 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. (3.4-4):

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

which is the average of the intensity levels of the pixels in the 3×3 neighborhood defined by the mask, as discussed earlier. Note that, instead of being 1/9,

$$\frac{1}{9}\times\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}\qquad \frac{1}{16}\times\begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix}$$

a b

FIGURE 3.32 Two 3×3 smoothing (averaging) filter masks. The constant multiplier in front of each mask is equal to 1 divided by the sum of the values of its coefficients, as is required to compute an average.

the coefficients of the filter are all 1s. 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 sometimes is called a *box filter*.

The second mask in Fig. 3.32 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. 3.32(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 the 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 chosen other weights to accomplish the same general objective. However, the sum of all the coefficients in the mask of Fig. 3.32(b) is equal to 16, an attractive feature for computer implementation because it is 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. 3.32, or similar arrangements, because the area spanned by these masks at any one location in an image is so small.

With reference to Eq. (3.4-1), 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

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

The parameters in this equation are as defined in Eq. (3.4-1). As before, it is understood that the complete filtered image is obtained by applying Eq. (3.5-1) for $x = 0, 1, 2, \ldots, M - 1$ and $y = 0, 1, 2, \ldots, N - 1$. The denominator in

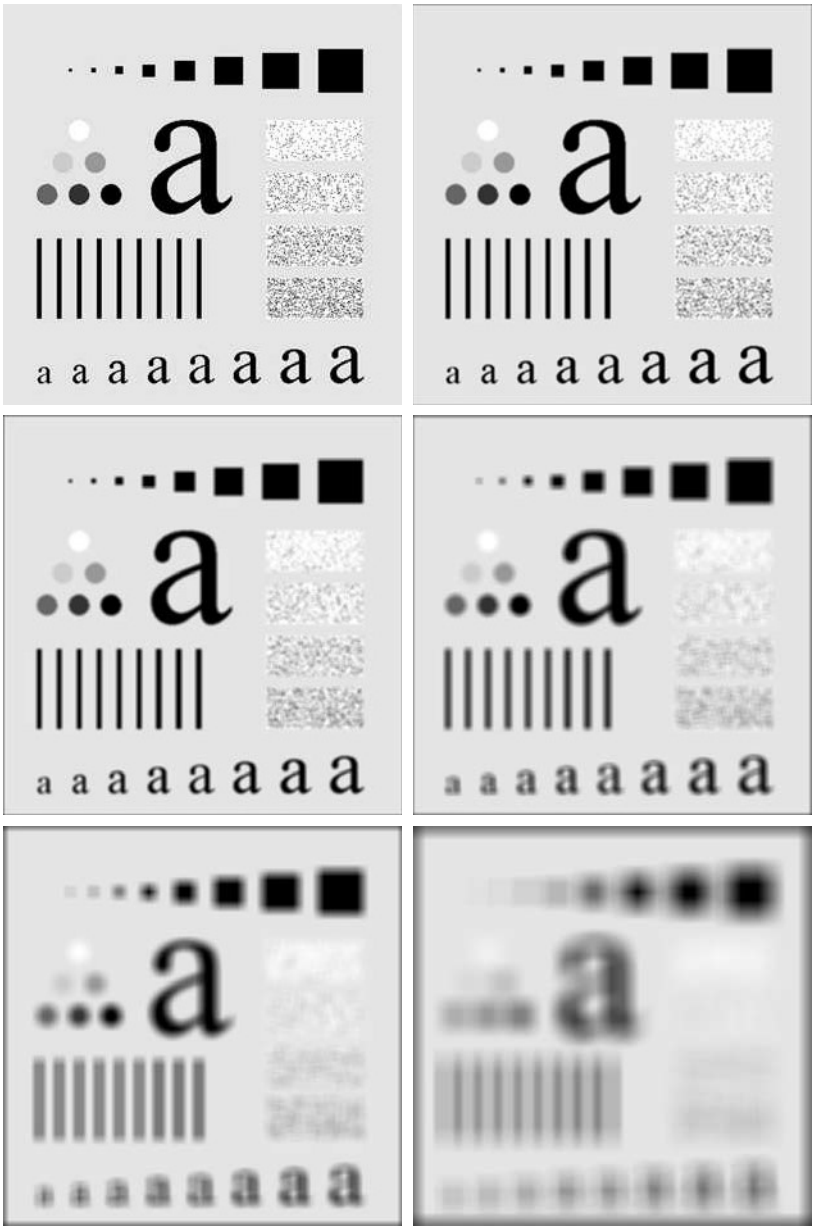
Eq. (3.5-1) is simply the sum of the mask coefficients and, therefore, it is a constant that needs to be computed only once.

EXAMPLE 3.13: Image smoothing with masks of various sizes.

■ The effects of smoothing as a function of filter size are illustrated in Fig. 3.33, which shows an original image and the corresponding smoothed results obtained using square averaging filters of sizes $m = 3, 5, 9, 15$, and 35 pixels, respectively. The principal features of these results are as follows: For $m = 3$, 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 black squares in the image, the small letter "a," and the fine grain noise show significant blurring when compared to the rest of the image. Note that the noise is less pronounced, and the jagged borders of the characters were pleasingly smoothed.

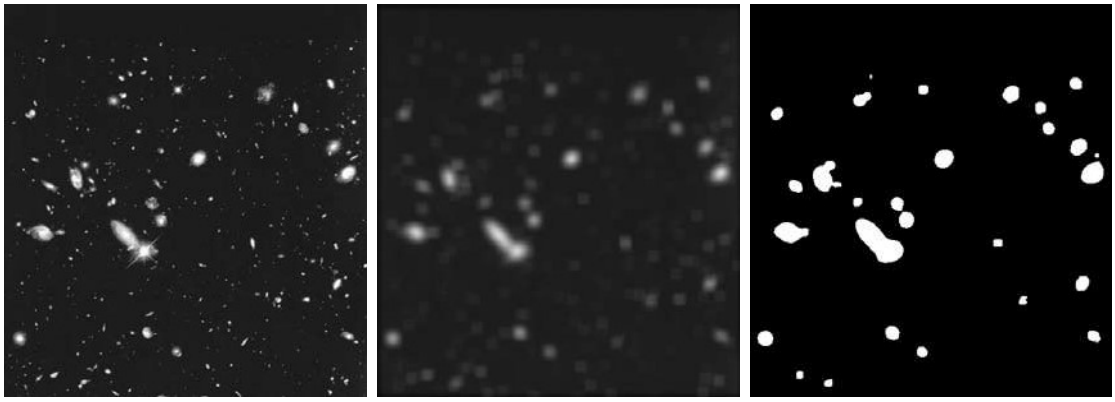
The result for $m = 5$ is somewhat similar, with a slight further increase in blurring. For $m = 9$ we see considerably more blurring, and the 20% black circle is not nearly as distinct from the background as in the previous three images, illustrating the blending effect that blurring has on objects whose intensities are close to that of its neighboring pixels. Note the significant further smoothing of the noisy rectangles. The results for $m = 15$ and 35 are extreme with respect to the sizes of the objects in the image. This type of aggresive blurring generally is 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. 3.33(f). Note also in this figure the pronounced black border. This is a result of padding the border of the original image with 0s (black) and then trimming off the padded area after filtering. 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 of 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. 3.34(a), which is an image from the Hubble telescope in orbit around the Earth. Figure 3.34(b) shows the result of applying a 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. 3.34(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.



a b
c d
e f

FIGURE 3.33 (a) Original image, of size 500 × 500 pixels. (b)–(f) Results of smoothing with square averaging filter masks of sizes $m = 3, 5, 9, 15$, 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 intensity 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.



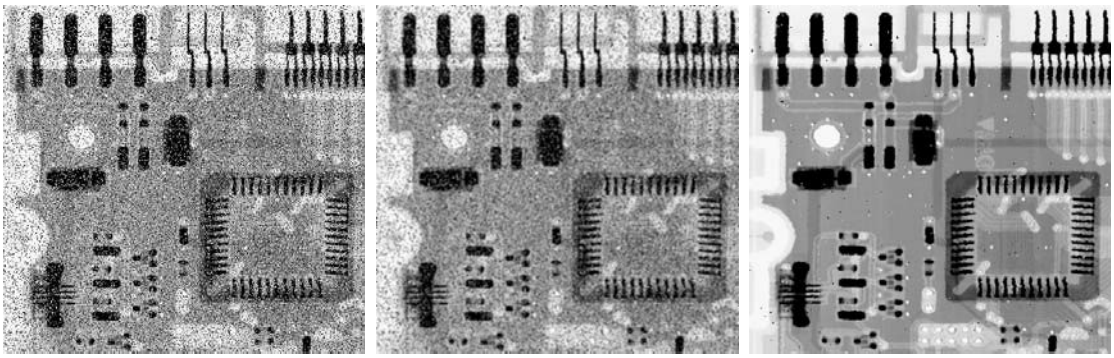
a b c

FIGURE 3.34 (a) Image of size 528 × 485 pixels from the Hubble Space Telescope. (b) Image filtered with a 15 × 15 averaging mask. (c) Result of thresholding (b). (Original image courtesy of NASA.)

3.5.2 Order-Statistic (Nonlinear) Filters

Order-statistic 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 filter in this category is the *median filter*, which, as its name implies, replaces the value of a pixel by the median of the intensity values 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 the neighborhood, determine their median, and assign that value to the corresponding pixel in the filtered image. For example, in a 3 × 3 neighborhood the median is the 5th largest value, in a 5 × 5 neighborhood it is 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 intensity 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 $m^2/2$ (one-half the filter area), are eliminated by an $m \times m$ median filter. In this case "eliminated" means forced to the median intensity of the neighbors. Larger clusters are affected considerably less.



a b c

FIGURE 3.35 (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.)

Although the median filter is by far the most useful order-statistic 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 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 for 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, min, and several other nonlinear filters are considered in more detail in Section 5.3.

See Section 10.3.5 regarding percentiles.

EXAMPLE 3.14: Use of median filtering for noise reduction.

■ Figure 3.35(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. 3.35(b) the result of processing the noisy image with a 3×3 neighborhood averaging mask, and in Fig. 3.35(c) the result of using a 3×3 median filter. The averaging filter blurred the image and its noise reduction performance was poor. 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 salt-and-pepper noise. ■

3.6 Sharpening Spatial Filters

The principal objective of sharpening is to highlight transitions in intensity. 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. Because averaging is analogous to integration, it is logical to conclude that sharpening can 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 intensity 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 intensities.

3.6.1 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 initially on one-dimensional derivatives. In particular, we are interested in the behavior of these derivatives in areas of constant intensity, at the onset and end of discontinuities (step and ramp discontinuities), and along intensity ramps. As you will see in Chapter 10, 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 areas of constant intensity; (2) must be nonzero at the onset of an intensity step or ramp; and (3) must be nonzero along ramps. Similarly, any definition of a *second derivative* (1) must be zero in constant areas; (2) must be nonzero at the onset *and* end of an intensity step or ramp; and (3) must be zero along ramps of constant slope. Because we are dealing with digital quantities whose values are finite, the maximum possible intensity 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

$$\frac{\partial f}{\partial x} = f(x + 1) - f(x) \tag{3.6-1}$$

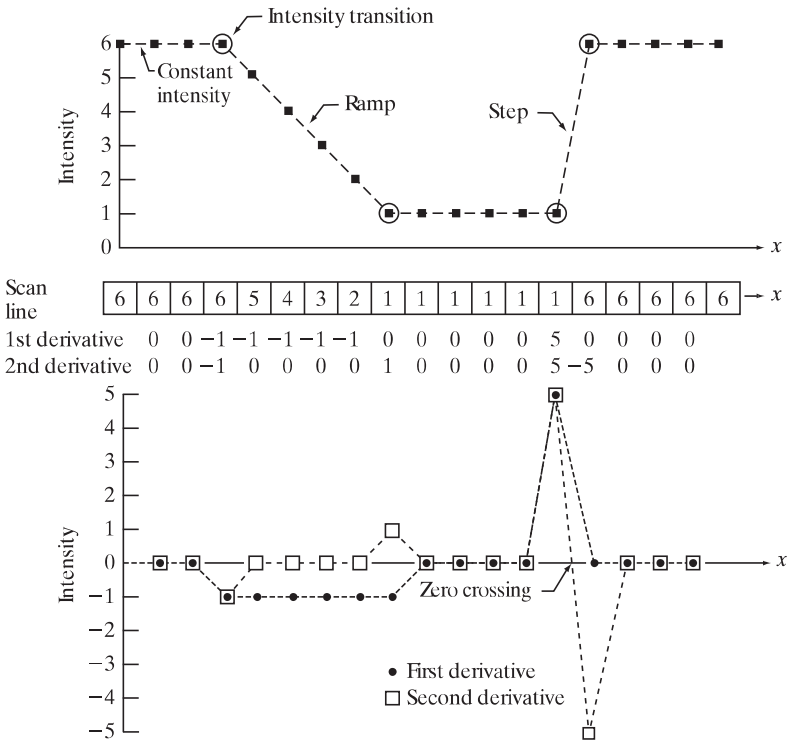
We return to Eq. (3.6-1) in Section 10.2.1 and show how it follows from a Taylor series expansion. For now, we accept it as a definition.

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. Clearly, $\partial f/\partial x = df/dx$ when there is only one variable in the function; the same is true for the second derivative.

We define the second-order derivative of $f(x)$ as the difference

$$\frac{\partial^2 f}{\partial x^2} = f(x + 1) + f(x - 1) - 2f(x) \tag{3.6-2}$$

It is easily verified that these two definitions satisfy the conditions stated above. To illustrate this, and to examine the similarities and differences between



a
b
c

FIGURE 3.36 Illustration of the first and second derivatives of a 1-D digital function representing a section of a horizontal intensity profile from an image. In (a) and (c) data points are joined by dashed lines as a visualization aid.

first- and second-order derivatives of a digital function, consider the example in Fig. 3.36.

Figure 3.36(b) (center of the figure) shows a section of a scan line (intensity profile). The values inside the small squares are the intensity values in the scan line, which are plotted as black dots above it in Fig. 3.36(a). The dashed line connecting the dots is included to aid visualization. As the figure shows, the scan line contains an intensity ramp, three sections of constant intensity, and an intensity step. The circles indicate the onset or end of intensity transitions. The first- and second-order derivatives computed using the two preceding definitions are included below the scan line in Fig. 3.36(b), and are plotted in Fig. 3.36(c). When computing the first derivative at a location x, we subtract the value of the function at that location from the next point. So this is a "look-ahead" operation. Similarly, to compute the second derivative at x, we use the previous and the next points in the computation. To avoid a situation in which the previous or next points are outside the range of the scan line, we show derivative computations in Fig. 3.36 from the second through the penultimate points in the sequence.

Let us consider the properties of the first and second derivatives as we traverse the profile from left to right. First, we encounter an area of constant intensity and, as Figs. 3.36(b) and (c) show, both derivatives are zero there, so condition (1) is satisfied for both. Next, we encounter an intensity ramp followed by a step, and we note that the first-order derivative is nonzero at the onset of the ramp and

the step; similarly, the second derivative is nonzero at the onset *and* end of both the ramp and the step; therefore, property (2) is satisfied for both derivatives. Finally, we see that property (3) is satisfied also for both derivatives because the first derivative is nonzero and the second is zero along the ramp. Note that the sign of the second derivative changes at the onset and end of a step or ramp. In fact, we see in Fig. 3.36(c) that in a step transition a line joining these two values crosses the horizontal axis midway between the two extremes. This *zero crossing* property is quite useful for locating edges, as you will see in Chapter 10.

Edges in digital images often are ramp-like transitions in intensity, in which case the first derivative of the image would result in thick edges because the derivative is nonzero along a ramp. On the other hand, the second derivative would produce a double edge one pixel thick, separated by zeros. From this, we conclude that the second derivative enhances fine detail much better than the first derivative, a property that is ideally suited for sharpening images. Also, as you will learn later in this section, second derivatives are much easier to implement than first derivates, so we focus our attention initially on second derivatives.

3.6.2 Using the Second Derivative for Image Sharpening—The Laplacian

In this section we consider the implementation of 2-D, second-order derivatives and their use for image sharpening. We return to this derivative in Chapter 10, where we use it extensively for image segmentation. 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.

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

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{3.6-3}$$

Because derivatives of any order are linear operations, the Laplacian is a linear operator. To express this equation in discrete form, we use the definition in Eq. (3.6-2), keeping in mind that we have to carry a second variable. In the x-direction, we have

$$\frac{\partial^2 f}{\partial x^2} = f(x + 1, y) + f(x - 1, y) - 2f(x, y) \tag{3.6-4}$$

and, similarly, in the y-direction we have

$$\frac{\partial^2 f}{\partial y^2} = f(x, y + 1) + f(x, y - 1) - 2f(x, y) \tag{3.6-5}$$

Therefore, it follows from the preceding three equations that the discrete Laplacian of two variables is

$$\nabla^2 f(x, y) = f(x + 1, y) + f(x - 1, y) + f(x, y + 1) + f(x, y - 1) - 4f(x, y) \tag{3.6-6}$$

This equation can be implemented using the filter mask in Fig. 3.37(a), which gives an isotropic result for rotations in increments of 90°. The mechanics of implementation are as in Section 3.5.1 for 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. (3.6-6), one for each of the two diagonal directions. The form of each new term is the same as either Eq. (3.6-4) or (3.6-5), but the coordinates are along the diagonals. Because each diagonal term also contains a $-2f(x, y)$ term, the total subtracted from the difference terms now would be $-8f(x, y)$. Figure 3.37(b) shows the filter mask used to implement this new definition. This mask yields isotropic results in increments of 45°. You are likely to see in practice the Laplacian masks in Figs. 3.37(c) and (d). They are obtained from definitions of the second derivatives that are the negatives of the ones we used in Eqs. (3.6-4) and (3.6-5). 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 intensity discontinuities in an image and deemphasizes regions with slowly varying intensity 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

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 3.37
(a) Filter mask used to implement Eq. (3.6-6).
(b) Mask used to implement an extension of this equation that includes the diagonal terms.
(c) and (d) Two other implementations of the Laplacian found frequently in practice.

effect of the Laplacian simply by adding the Laplacian image to the original. 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 sharpening is

$$g(x, y) = f(x, y) + c\left[\nabla^2 f(x, y)\right] \tag{3.6-7}$$

where $f(x, y)$ and $g(x, y)$ are the input and sharpened images, respectively. The constant is $c = -1$ if the Laplacian filters in Fig. 3.37(a) or (b) are used, and $c = 1$ if either of the other two filters is used.

EXAMPLE 3.15: Image sharpening using the Laplacian.

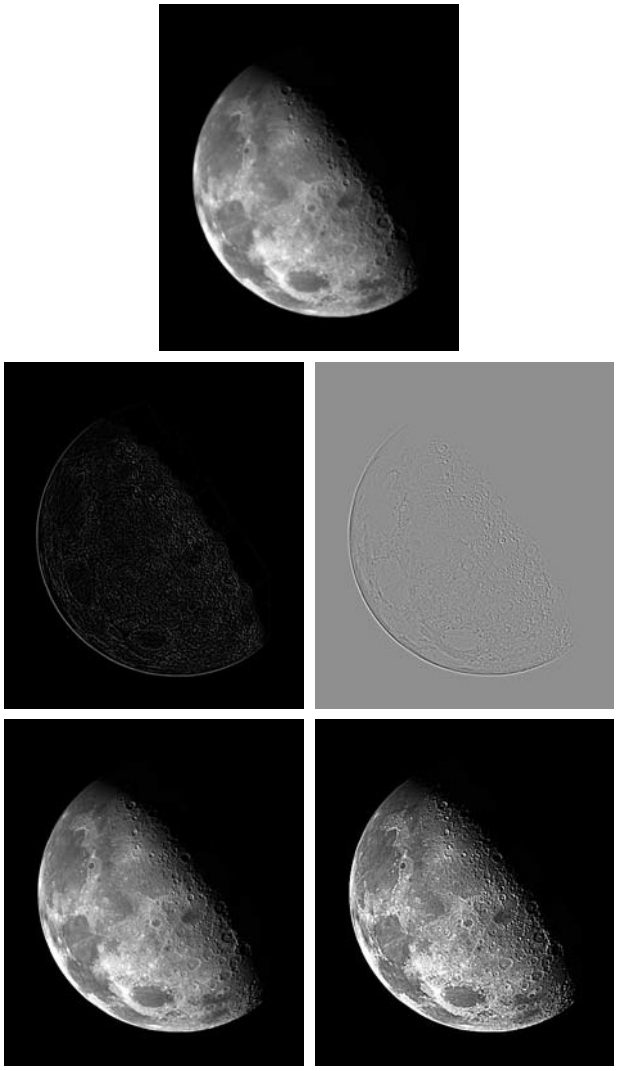
■ Figure 3.38(a) shows a slightly blurred image of the North Pole of the moon. Figure 3.38(b) shows the result of filtering this image with the Laplacian mask in Fig. 3.37(a). Large sections of this image are black because the Laplacian contains both positive and negative values, and all negative values are clipped at 0 by the display.

A typical way to scale a Laplacian image is to add to it its minimum value to bring the new minimum to zero and then scale the result to the full $[0, L - 1]$ intensity range, as explained in Eqs. (2.6-10) and (2.6-11). The image in Fig. 3.38(c) was scaled in this manner. Note that the dominant features of the image are edges and sharp intensity discontinuities. The background, previously black, is now gray due to scaling. This grayish appearance is typical of Laplacian images that have been scaled properly. Figure 3.38(d) shows the result obtained using Eq. (3.6-7) with $c = -1$. The detail in this image is unmistakably clearer and sharper than in the original image. Adding the original image to the Laplacian restored the overall intensity variations in the image, with the Laplacian increasing the contrast at the locations of intensity discontinuities. The net result is an image in which small details were enhanced and the background tonality was reasonably preserved. Finally, Fig. 3.38(e) shows the result of repeating the preceding procedure with the filter in Fig. 3.37(b). Here, we note a significant improvement in sharpness over Fig. 3.38(d). This is not unexpected because using the filter in Fig. 3.37(b) provides additional differentiation (sharpening) in the diagonal directions. Results such as those in Figs. 3.38(d) and (e) have made the Laplacian a tool of choice for sharpening digital images. ■

3.6.3 Unsharp Masking and Highboost Filtering

A process that has been used for many years by the printing and publishing industry to sharpen images consists of subtracting an unsharp (smoothed) version of an image from the original image. This process, called *unsharp masking*, consists of the following steps:

1. Blur the original image.
2. Subtract the blurred image from the original (the resulting difference is called the *mask*.)
3. Add the mask to the original.



a
b c
d e

FIGURE 3.38
(a) Blurred image of the North Pole of the moon. (b) Laplacian without scaling. (c) Laplacian with scaling. (d) Image sharpened using the mask in Fig. 3.37(a). (e) Result of using the mask in Fig. 3.37(b). (Original image courtesy of NASA.)

Letting $\overline{f}(x, y)$ denote the blurred image, unsharp masking is expressed in equation form as follows. First we obtain the mask:

$$g_{\text{mask}}(x, y) = f(x, y) - \overline{f}(x, y) \tag{3.6-8}$$

Then we add a weighted portion of the mask back to the original image:

$$g(x, y) = f(x, y) + k * g_{\text{mask}}(x, y) \tag{3.6-9}$$

where we included a weight, k ($k \geq 0$), for generality. When $k = 1$, we have unsharp masking, as defined above. When $k > 1$, the process is referred to as

a
b
c
d

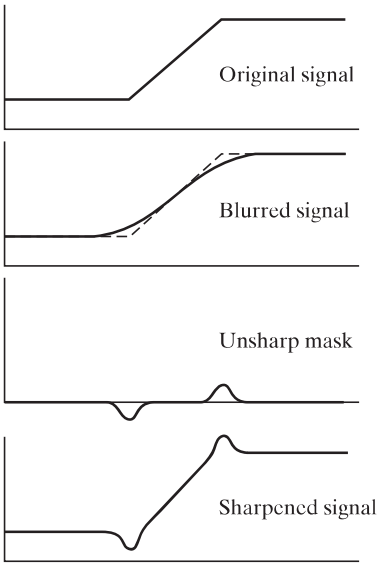


FIGURE 3.39 1-D illustration of the mechanics of unsharp masking. (a) Original signal. (b) Blurred signal with original shown dashed for reference. (c) Unsharp mask. (d) Sharpened signal, obtained by adding (c) to (a).

highboost filtering. Choosing $k < 1$ de-emphasizes the contribution of the unsharp mask.

Figure 3.39 explains how unsharp masking works. The intensity profile in Fig. 3.39(a) can be interpreted as a horizontal scan line through a vertical edge that transitions from a dark to a light region in an image. Figure 3.39(b) shows the result of smoothing, superimposed on the original signal (shown dashed) for reference. Figure 3.39(c) is the unsharp mask, obtained by subtracting the blurred signal from the original. By comparing this result with the section of Fig. 3.36(c) corresponding to the ramp in Fig. 3.36(a), we note that the unsharp mask in Fig. 3.39(c) is very similar to what we would obtain using a second-order derivative. Figure 3.39(d) is the final sharpened result, obtained by adding the mask to the original signal. The points at which a change of slope in the intensity occurs in the signal are now emphasized (sharpened). Observe that negative values were added to the original. Thus, it is possible for the final result to have negative intensities if the original image has any zero values or if the value of k is chosen large enough to emphasize the peaks of the mask to a level larger than the minimum value in the original. Negative values would cause a dark halo around edges, which, if k is large enough, can produce objectionable results.

EXAMPLE 3.16: Image sharpening using unsharp masking.

■ Figure 3.40(a) shows a slightly blurred image of white text on a dark gray background. Figure 3.40(b) was obtained using a Gaussian smoothing filter (see Section 3.4.4) of size 5×5 with $\sigma = 3$. Figure 3.40(c) is the unsharp mask, obtained using Eq. (3.6-8). Figure 3.40(d) was obtained using unsharp

a
b
c
d
e

FIGURE 3.40
(a) Original image. (b) Result of blurring with a Gaussian filter. (c) Unsharp mask. (d) Result of using unsharp masking. (e) Result of using highboost filtering.

masking [Eq. (3.6-9) with $k = 1$]. This image is a slight improvement over the original, but we can do better. Figure 3.40(e) shows the result of using Eq. (3.6-9) with $k = 4.5$, the largest possible value we could use and still keep positive all the values in the final result. The improvement in this image over the original is significant. ■

3.6.4 Using First-Order Derivatives for (Nonlinear) Image Sharpening—The Gradient

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

$$\nabla f \equiv \text{grad}(f) \equiv \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f}{\partial x} \\ \dfrac{\partial f}{\partial y} \end{bmatrix} \tag{3.6-10}$$

We discuss the gradient in detail in Section 10.2.5. Here, we are interested only in using the magnitude of the gradient for image sharpening.

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

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

$$M(x, y) = \text{mag}(\nabla f) = \sqrt{g_x^2 + g_y^2} \tag{3.6-11}$$

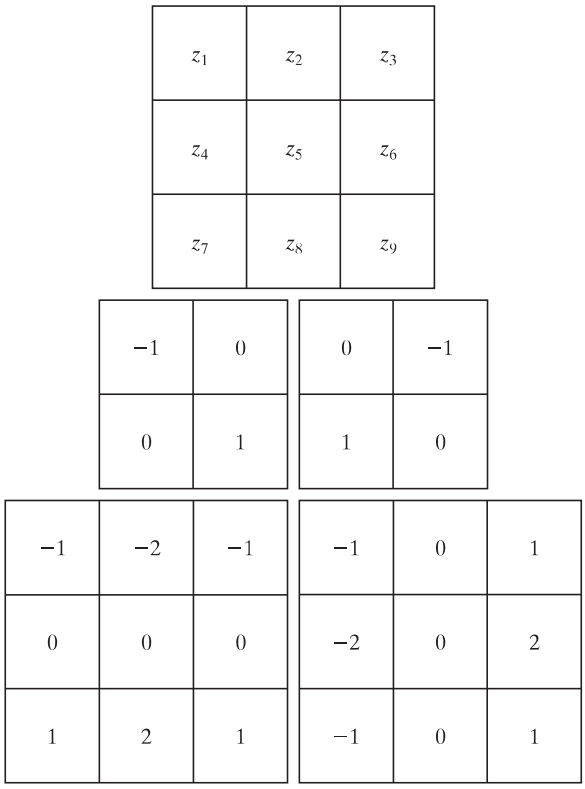
is the *value* at (x, y) of the rate of change in the direction of the gradient vector. Note that $M(x, y)$ is an image of the same size as the original, created when x and y are allowed to vary over all pixel locations in f. It is common practice to refer to this image as the *gradient image* (or simply as the *gradient* when the meaning is clear).

Because the components of the gradient vector are derivatives, they are linear operators. However, the magnitude of this vector is not because of the squaring and square root operations. On the other hand, the partial derivatives in Eq. (3.6-10) are not rotation invariant (isotropic), but the magnitude of the gradient vector is. In some implementations, it is more suitable computationally to approximate the squares and square root operations by absolute values:

$$M(x, y) \approx |g_x| + |g_y| \tag{3.6-12}$$

This expression still preserves the relative changes in intensity, but the isotropic property is lost in general. However, as in the case of the Laplacian, the isotropic properties of the discrete gradient defined in the following paragraph are preserved only for a limited number of rotational increments that depend on the filter masks used to approximate the derivatives. As it turns out, the most popular masks used to approximate the gradient are isotropic at multiples of 90°. These results are independent of whether we use Eq. (3.6-11) or (3.6-12), so nothing of significance is lost in using the latter equation if we choose to do so.

As in the case of the Laplacian, we now define discrete 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. 3.41(a) to denote the intensities of image points in a 3×3 region. For



a
b c
d e

FIGURE 3.41
A 3×3 region of an image (the zs are intensity values). (b)–(c) Roberts cross gradient operators. (d)–(e) Sobel operators. All the mask coefficients sum to zero, as expected of a derivative operator.

example, the center point, z_5, denotes $f(x, y)$ at an arbitrary location, (x, y); z_1 denotes $f(x - 1, y - 1)$; and so on, using the notation introduced in Fig. 3.28. As indicated in Section 3.6.1, 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:

$$g_x = (z_9 - z_5) \quad \text{and} \quad g_y = (z_8 - z_6) \tag{3.6-13}$$

If we use Eqs. (3.6-11) and (3.6-13), we compute the gradient image as

$$M(x, y) = \left[(z_9 - z_5)^2 + (z_8 - z_6)^2\right]^{1/2} \tag{3.6-14}$$

If we use Eqs. (3.6-12) and (3.6-13), then

$$M(x, y) \approx |z_9 - z_5| + |z_8 - z_6| \tag{3.6-15}$$

where it is understood that x and y vary over the dimensions of the image in the manner described earlier. The partial derivative terms needed in equation (3.6-13) can be implemented using the two linear filter masks in Figs. 3.41(b) and (c). These masks are referred to as the *Roberts cross-gradient operators*.

Masks of even sizes are awkward to implement because they do not have a center of symmetry. The smallest filter masks in which we are interested are of size 3×3. Approximations to g_x and g_y using a 3×3 neighborhood centered on z_5 are as follows:

$$g_x = \frac{\partial f}{\partial x} = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3) \tag{3.6-16}$$

and

$$g_y = \frac{\partial f}{\partial y} = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7) \tag{3.6-17}$$

These equations can be implemented using the masks in Figs. 3.41(d) and (e). The difference between the third and first rows of the 3×3 image region implemented by the mask in Fig. 3.41(d) approximates the partial derivative in the x-direction, and the difference between the third and first columns in the other mask approximates the derivative in the y-direction. After computing the partial derivatives with these masks, we obtain the magnitude of the gradient as before. For example, substituting g_x and g_y into Eq. (3.6-12) yields

$$\begin{aligned} M(x, y) \approx\ & |(z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)| \\ & + |(z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)| \end{aligned} \tag{3.6-18}$$

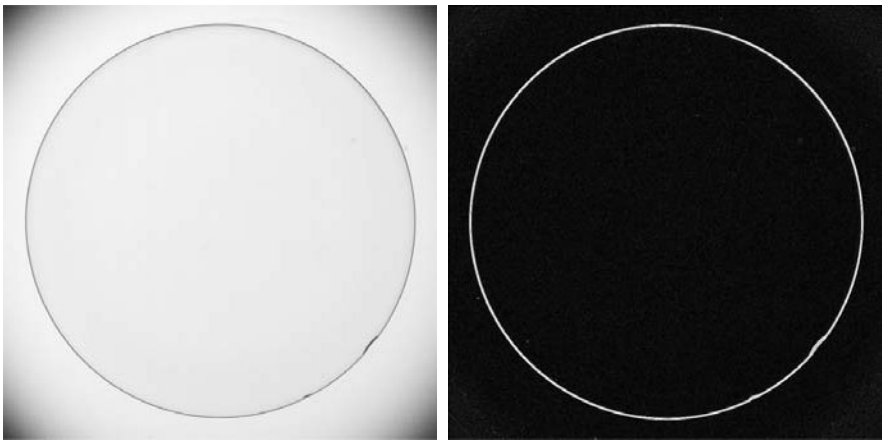
The masks in Figs. 3.41(d) and (e) are called the *Sobel operators*. The idea behind using a weight value of 2 in the center coefficient is to achieve some smoothing by giving more importance to the center point (we discuss this in more detail in Chapter 10). Note that the coefficients in all the masks shown in Fig. 3.41 sum to 0, indicating that they would give a response of 0 in an area of constant intensity, as is expected of a derivative operator.

As mentioned earlier, the computations of g_x and g_y are linear operations because they involve derivatives and, therefore, can be implemented as a sum of products using the spatial masks in Fig. 3.41. The nonlinear aspect of sharpening with the gradient is the computation of $M(x, y)$ involving squaring and square roots, or the use of absolute values, all of which are nonlinear operations. These operations are performed *after* the linear process that yields g_x and g_y.

EXAMPLE 3.17: 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 example, enhancement is used as a preprocessing step for automated inspection, rather than for human analysis.

Figure 3.42(a) shows an optical image of a contact lens, illuminated by a lighting arrangement designed to highlight imperfections, such as the two edge defects in the lens boundary seen at 4 and 5 o'clock. Figure 3.42(b) shows the gradient obtained using Eq. (3.6-12) with the two Sobel masks in Figs. 3.41(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. The gradient can be used also to highlight small specs that may not be readily visible in a 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. ■



a b

FIGURE 3.42 (a) Optical image of contact lens (note defects on the boundary at 4 and 5 o'clock). (b) Sobel gradient. (Original image courtesy of Pete Sites, Perceptics Corporation.)

3.7 Combining Spatial Enhancement Methods

With a few exceptions, like combining blurring with thresholding (Fig. 3.34), we have focused attention thus far on individual approaches. Frequently, a given task will require application of several complementary 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 thus far in this chapter to address a difficult image enhancement task.

The image in Fig. 3.43(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 intensity 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 Fig. 2.30 regarding masking). Finally, we will attempt to increase the dynamic range of the intensity levels by using an intensity transformation.

Figure 3.43(b) shows the Laplacian of the original image, obtained using the filter in Fig. 3.37(d). This image was scaled (for display only) using the same technique as in Fig. 3.38(c). We can obtain a sharpened image at this point simply by adding Figs. 3.43(a) and (b), according to Eq. (3.6-7). Just by looking at the noise level in Fig. 3.43(b), we would expect a rather noisy sharpened image if we added Figs. 3.43(a) and (b), a fact that is confirmed by the result in Fig. 3.43(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 3.6.1. 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 average response in areas of significant intensity transitions (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 interpreted 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.

Figure 3.43(d) shows the Sobel gradient of the original image, computed using Eq. (3.6-12). Components g_x and g_y were obtained using the masks in Figs. 3.41(d) and (e), respectively. As expected, edges are much more dominant

a b
c d

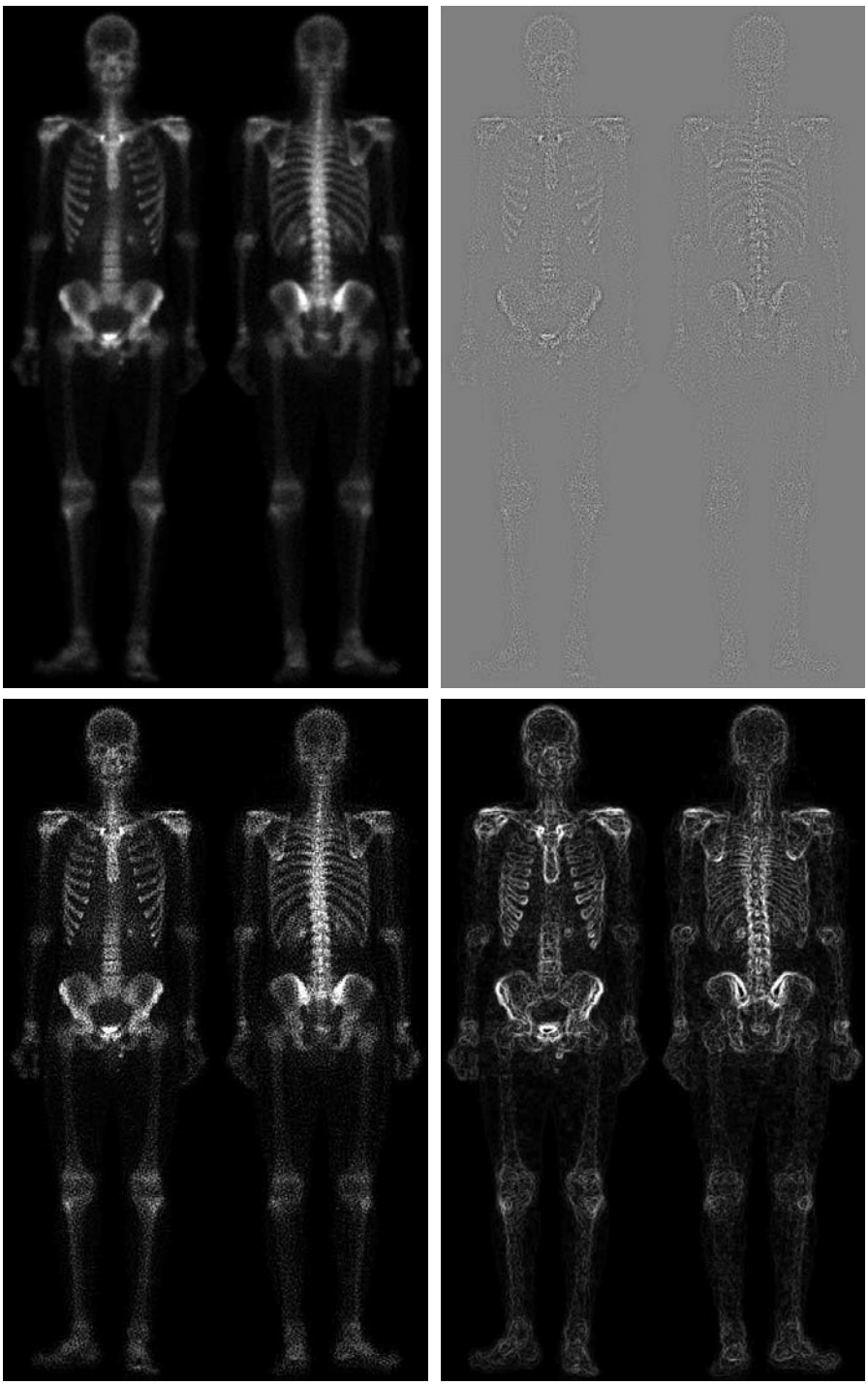
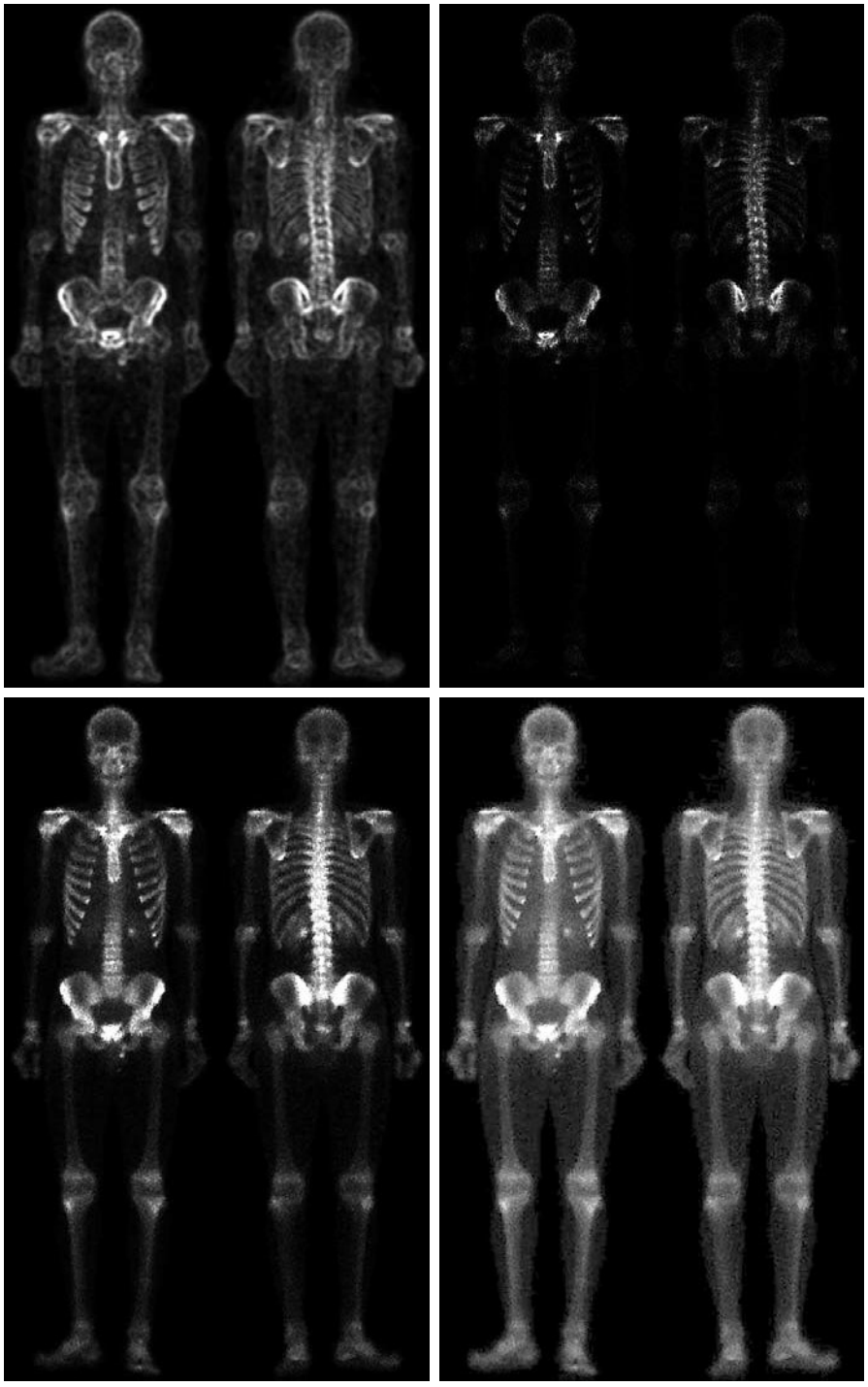


FIGURE 3.43
(a) Image of whole body bone scan.
(b) Laplacian of (a). (c) Sharpened image obtained by adding (a) and (b). (d) Sobel gradient of (a).



e f
g h

FIGURE 3.43
(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.)

in this image than in the Laplacian image. The smoothed gradient image in Fig. 3.43(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 Laplacian image. 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. 3.43(d) and (e) are much brighter than Fig. 3.43(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. 3.43(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. 3.43(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 cord, pelvis, and skull. This type of improvement would not have been possible by using the Laplacian or the gradient alone.

The sharpening procedure just discussed does not affect in an appreciable way the dynamic range of the intensity 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 intensity 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 intensity 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 intensity 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 in Fig. 3.43(h), obtained with $\gamma = 0.5$ and $c = 1$. Comparing this image with Fig. 3.43(g), we see that significant new detail is visible in Fig. 3.43(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 intensity levels also enhanced noise, but Fig. 3.43(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 example 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.

3.8 Using Fuzzy Techniques for Intensity Transformations and Spatial Filtering

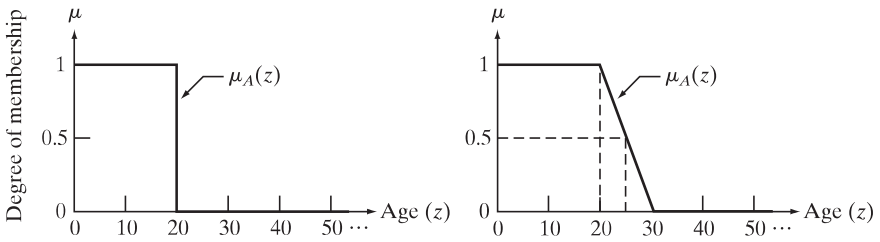
We conclude this chapter with an introduction to fuzzy sets and their application to intensity transformations and spatial filtering, which are the main topics of discussion in the preceding sections. As it turns out, these two applications are among the most frequent areas in which fuzzy techniques for image processing are applied. The references at the end of this chapter provide an entry point to the literature on fuzzy sets and to other applications of fuzzy techniques in image processing. As you will see in the following discussion, fuzzy sets provide a framework for incorporating human knowledge in the solution of problems whose formulation is based on imprecise concepts.

3.8.1 Introduction

As noted in Section 2.6.4, a *set* is a collection of objects (elements) and *set theory* is the set of tools that deals with operations on and among sets. Set theory, along with mathematical logic, is one of the axiomatic foundations of classical mathematics. Central to set theory is the notion of set membership. We are used to dealing with so-called "crisp" sets, whose membership only can be true or false in the traditional sense of bi-valued Boolean logic, with 1 typically indicating true and 0 indicating false. For example, let Z denote the set of all people, and suppose that we want to define a subset, A, of Z, called the "set of young people." In order to form this subset, we need to define a *membership function* that assigns a value of 1 or 0 to every element, z, of Z. Because we are dealing with a bi-valued logic, the membership function simply defines a threshold at or below which a person is considered young, and above which a person is considered not young. Figure 3.44(a) summarizes this concept using an age threshold of 20 years and letting $\mu_A(z)$ denote the membership function just discussed.

Membership functions also are called *characteristic functions.*

We see an immediate difficulty with this formulation: A person 20 years of age is considered young, but a person whose age is 20 years and 1 second is not a member of the set of young people. This is a fundamental problem with crisp sets that limits the use of classical set theory in many practical applications.



a b

FIGURE 3.44 Membership functions used to generate (a) a crisp set, and (b) a fuzzy set.

What we need is more flexibility in what we mean by "young," that is, a *gradual* transition from young to not young. Figure 3.44(b) shows one possibility. The key feature of this function is that it is infinite valued, thus allowing a continuous transition between young and not young. This makes it possible to have *degrees* of "youngness." We can make statements now such as a person being young (upper flat end of the curve), relatively young (toward the beginning of the ramp), 50% young (in the middle of the ramp), not so young (toward the end of the ramp), and so on (note that decreasing the slope of the curve in Fig. 3.44(b) introduces more vagueness in what we mean by "young.") These types of vague (*fuzzy*) statements are more in line with what humans use when talking imprecisely about age. Thus, we may interpret infinite-valued membership functions as being the foundation of a *fuzzy logic*, and the sets generated using them may be viewed as *fuzzy sets*. These ideas are formalized in the following section.

3.8.2 Principles of Fuzzy Set Theory

Fuzzy set theory was introduced by L. A. Zadeh in a paper more than four decades ago (Zadeh [1965]). As the following discussion shows, fuzzy sets provide a formalism for dealing with imprecise information.

Definitions

We follow conventional fuzzy set notation in using Z, instead of the more traditional set notation U, to denote the set universe in a given application.

Let Z be a set of elements (objects), with a generic element of Z denoted by z; that is, $Z = \{z\}$. This set is called the *universe of discourse*. A *fuzzy set*[†] A in Z is characterized by a *membership function*, $\mu_A(z)$, that associates with each element of Z a real number in the interval $[0, 1]$. The value of $\mu_A(z)$ at z represents the *grade of membership* of z in A. The nearer the value of $\mu_A(z)$ is to unity, the higher the membership grade of z in A, and conversely when the value of $\mu_A(z)$ is closer to zero. The concept of "belongs to," so familiar in ordinary sets, does not have the same meaning in fuzzy set theory. With ordinary sets, we say that an element either belongs or does not belong to a set. With fuzzy sets, we say that all zs for which $\mu_A(z) = 1$ are *full* members of the set, all zs for which $\mu_A(z) = 0$ are *not* members of the set, and all zs for which $\mu_A(z)$ is between 0 and 1 have *partial* membership in the set. Therefore, a fuzzy set is an *ordered pair* consisting of values of z and a corresponding membership function that assigns a grade of membership to each z. That is,

$$A = \{z, \mu_A(z) | z \in Z\} \tag{3.8-1}$$

When the variables are continuous, the set A in this equation can have an infinite number of elements. When the values of z are discrete, we can show the elements of A explicitly. For instance, if age increments in Fig. 3.44 were limited to integer years, then we would have

$$A = \{(1, 1), (2, 1), (3, 1), \ldots, (20, 1), (21, 0.9), (22, 0.8), \ldots, (25, 0.5)(24, 0.4), \ldots, (29, 0.1)\}$$

[†]The term *fuzzy subset* is also used in the literature, indicating that A is as subset of Z. However, *fuzzy set* is used more frequently.

where, for example, the element (22, 0.8) denotes that age 22 has a 0.8 degree of membership in the set. All elements with ages 20 and under are full members of the set and those with ages 30 and higher are not members of the set. Note that a plot of this set would simply be discrete points lying on the curve of Fig. 3.44(b), so $\mu_A(z)$ completely defines A. Viewed another way, we see that a (discrete) fuzzy set is nothing more than the set of points of a function that maps each element of the problem domain (universe of discourse) into a number greater than 0 and less than or equal to 1. Thus, one often sees the terms *fuzzy set* and *membership function* used interchangeably.

When $\mu_A(z)$ can have only two values, say 0 and 1, the membership function reduces to the familiar characteristic function of an ordinary (crisp) set A. Thus, ordinary sets are a special case of fuzzy sets. Next, we consider several definitions involving fuzzy sets that are extensions of the corresponding definitions from ordinary sets.

Empty set: A fuzzy set is *empty* if and only if its membership function is identically zero in Z.

Equality: Two fuzzy sets A and B are *equal*, written $A = B$, if and only if $\mu_A(z) = \mu_B(z)$ for all $z \in Z$.

The notation "for all $z \in Z$" reads: "for all z belonging to Z."

Complement: The *complement* (NOT) of a fuzzy set A, denoted by $\overline{A}$, or NOT(A), is defined as the set whose membership function is

$$\mu_{\overline{A}}(z) = 1 - \mu_A(z) \tag{3.8-2}$$

for all $z \in Z$.

Subset: A fuzzy set A is a *subset* of a fuzzy set B if and only if

$$\mu_A(z) \leq \mu_B(z) \tag{3.8-3}$$

for all $z \in Z$.

Union: The *union* (OR) of two fuzzy sets A and B, denoted $A \cup B$, or A OR B, is a fuzzy set U with membership function

$$\mu_U(z) = \max[\mu_A(z), \mu_B(z)] \tag{3.8-4}$$

for all $z \in Z$.

Intersection: The *intersection* (AND) of two fuzzy sets A and B, denoted $A \cap B$, or A AND B, is a fuzzy set I with membership function

$$\mu_I(z) = \min[\mu_A(z), \mu_B(z)] \tag{3.8-5}$$

for all $z \in Z$.

Note that the familiar terms NOT, OR, and AND are used interchangeably when working with fuzzy sets to denote complementation, union, and intersection, respectively.

a b
c d

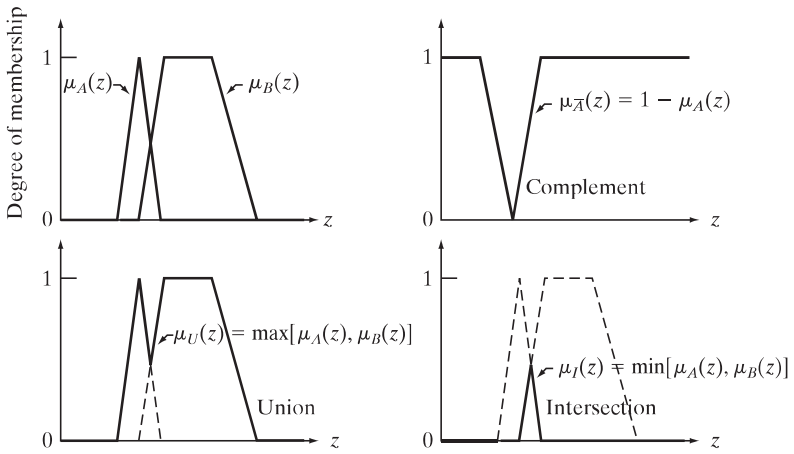


FIGURE 3.45 (a) Membership functions of two sets, A and B. (b) Membership function of the complement of A. (c) and (d) Membership functions of the union and intersection of the two sets.

EXAMPLE 3.18: Illustration of fuzzy set definitions.

■ Figure 3.45 illustrates some of the preceding definitions. Figure 3.45(a) shows the membership functions of two sets, A and B, and Fig. 3.45(b) shows the membership function of the complement of A. Figure 3.45(c) shows the membership function of the union of A and B, and Fig. 3.45(d) shows the corresponding result for the intersection of these two sets. Note that these figures are consistent with our familiar notion of complement, union, and intersection of crisp sets.[†] ■

Although fuzzy logic and probability operate over the same $[0, 1]$ interval, there is a significant distinction to be made between the two. Consider the example from Fig. 3.44. A probabilistic statement might read: "There is a 50% chance that a person is young," while a fuzzy statement would read "A person's degree of membership within the set of young people is 0.5." The difference between these two statements is important. In the first statement, a person is considered to be either in the set of young or the set of not young people; we simply have only a 50% chance of knowing to which set the person belongs. The second statement presupposes that a person is young to some degree, with that degree being in this case 0.5. Another interpretation is to say that this is an "average" young person: not really young, but not too near being not young. In other words, fuzzy logic is not probabilistic at all; it just deals with degrees of membership in a set. In this sense, we see that fuzzy logic concepts find application in situations characterized by vagueness and imprecision, rather than by randomness.

[†]You are likely to encounter examples in the literature in which the area under the curve of the membership function of, say, the intersection of two fuzzy sets, is shaded to indicate the result of the operation. This is a carryover from ordinary set operations and is incorrect. Only the points along the membership function itself are applicable when dealing with fuzzy sets.

Some common membership functions

Types of membership functions used in practice include the following.

Triangular:

$$\mu(z) = \begin{cases} 1 - (a - z)/b & a - b \le z < a \\ 1 - (z - a)/c & a \le z \le a + c \\ 0 & \text{otherwise} \end{cases} \tag{3.8-6}$$

Trapezoidal:

$$\mu(z) = \begin{cases} 1 - (a - z)/c & a - c \le z < a \\ 1 & a \le z < b \\ 1 - (z - b)/d & b \le z \le b + d \\ 0 & \text{otherwise} \end{cases} \tag{3.8-7}$$

Sigma:

$$\mu(z) = \begin{cases} 1 - (a - z)/b & a - b \le z \le a \\ 1 & z > a \\ 0 & \text{otherwise} \end{cases} \tag{3.8-8}$$

S-shape:

$$S(z; a, b, c) = \begin{cases} 0 & z < a \\ 2\left(\dfrac{z - a}{c - a}\right)^2 & a \le z \le b \\ 1 - 2\left(\dfrac{z - c}{c - a}\right)^2 & b < z \le c \\ 1 & z > c \end{cases} \tag{3.8-9}$$

Bell-shape:

$$\mu(z) = \begin{cases} S(z; c - b, c - b/2, c) & z \le c \\ 1 - S(z; c, c + b/2, c + b) & z > c \end{cases} \tag{3.8-10}$$

The bell-shape function sometimes is referred to as the Π (or π) function.

Truncated Gaussian:

$$\mu(z) = \begin{cases} e^{-\frac{(z-a)^2}{2b^2}} & a - c \le z \le a + c \\ 0 & \text{otherwise} \end{cases} \tag{3.8-11}$$

Typically, only the independent variable, z, is included when writing $\mu(z)$ in order to simplify equations. We made an exception in Eq. (3.8-9) in order to use its form in Eq. (3.8-10). Figure 3.46 shows examples of the membership

a b
c d
e f

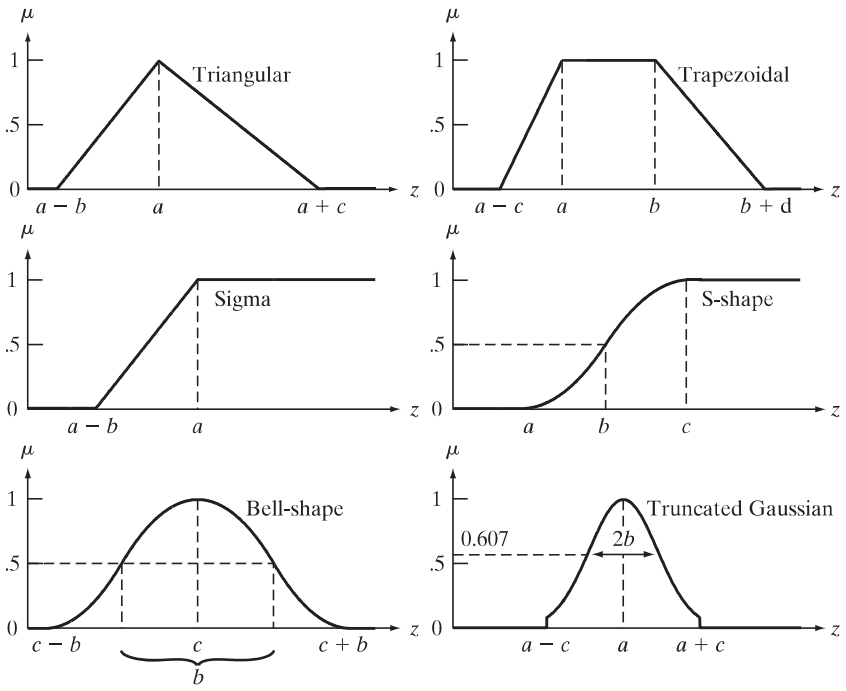


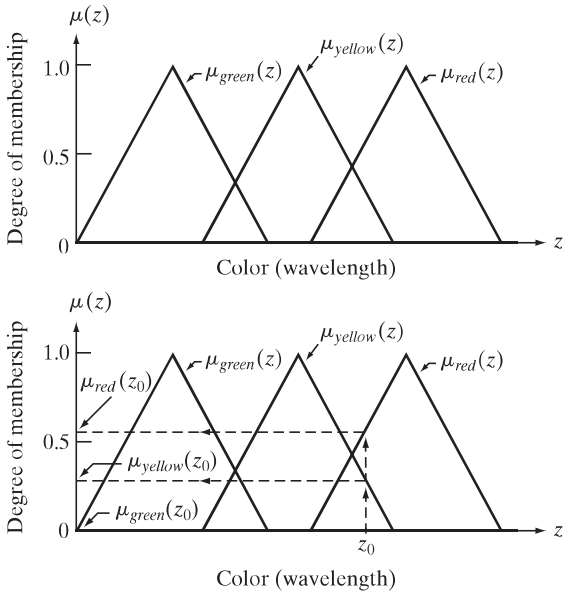
FIGURE 3.46 Membership functions corresponding to Eqs. (3.8-6)–(3.8-11).

functions just discussed. The first three functions are piecewise linear, the next two functions are smooth, and the last function is a truncated Gaussian function. Equation (3.8-9) describes an important S-shape function that it used frequently when working with fuzzy sets. The value of $z = b$ at which $S = 0.5$ in this equation is called the *crossover point*. As Fig. 3.46(d) shows, this is the point at which the curve changes inflection. It is not difficult to show (Problem 3.31) that $b = (a + c)/2$. In the bell-shape curve of Fig. 3.46(e), the value of b defines the *bandwidth* of the curve.

3.8.3 Using Fuzzy Sets

In this section, we lay the foundation for using fuzzy sets and illustrate the resulting concepts with examples from simple, familiar situations. We then apply the results to image processing in Sections 3.8.4 and 3.8.5. Approaching the presentation in this way makes the material much easier to understand, especially for readers new to this area.

Suppose that we are interested in using color to categorize a given type of fruit into three groups: verdant, half-mature, and mature. Assume that observations of fruit at various stages of maturity have led to the conclusion that verdant fruit is green, half-mature fruit is yellow, and mature fruit is red. The labels *green*, *yellow*, and *red* are vague descriptions of color sensation. As a starting point, these labels have to be expressed in a fuzzy format. That is, they have to be *fuzzified*. This is achieved by defining membership as a function of



a
b

FIGURE 3.47
(a) Membership functions used to fuzzify color. (b) Fuzzifying a specific color z_0. (Curves describing color sensation are bell shaped; see Section 6.1 for an example. However, using triangular shapes as an approximation is common practice when working with fuzzy sets.)

color (wavelength of light), as Fig. 3.47(a) shows. In this context, *color* is a *linguistic variable*, and a particular color (e.g., *red* at a fixed wavelength) is a *linguistic value*. A linguistic value, z_0, is fuzzified by using a membership functions to map it to the interval $[0, 1]$, as Fig. 3.47(b) shows.

The problem-specific *knowledge* just explained can be formalized in the form of the following *fuzzy IF-THEN rules*:

R_1: IF the color is *green*, THEN the fruit is *verdant*.

OR

R_2: IF the color is *yellow*, THEN the fruit is *half-mature*.

OR

R_3: IF the color is *red*, THEN the fruit is *mature*.

The part of an IF-THEN rule to the left of THEN often is referred to as the *antecedent* (or *premise*). The part to the right is called the *consequent* (or *conclusion*.)

These rules represent the sum total of our knowledge about this problem; they are really nothing more than a formalism for a thought process.

The next step of the procedure is to find a way to use inputs (color) and the knowledge base represented by the IF-THEN rules to create the output of the fuzzy system. This process is known as *implication* or *inference*. However, before implication can be applied, the antecedent of each rule has to be processed to yield a *single* value. As we show at the end of this section, multiple parts of an antecedent are linked by ANDs and ORs. Based on the definitions from Section 3.8.2, this means performing min and max operations. To simplify the explanation, we deal initially with rules whose antecedents contain only one part.

Because we are dealing with fuzzy inputs, the outputs themselves are fuzzy, so membership functions have to be defined for the outputs as well. Figure 3.48

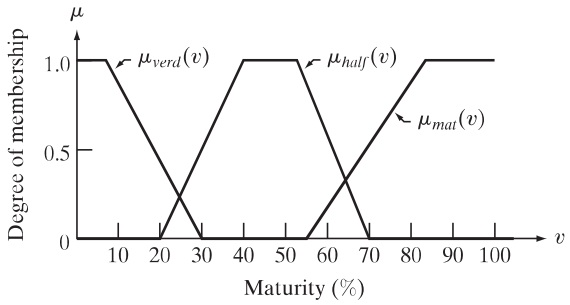


FIGURE 3.48 Membership functions characterizing the outputs *verdant*, *half-mature*, and *mature*.

shows the membership functions of the fuzzy outputs we are going to use in this example. Note that the independent variable of the outputs is *maturity*, which is different from the independent variable of the inputs.

Figures 3.47 and 3.48, together with the rule base, contain all the information required to relate inputs and outputs. For example, we note that the expression *red* AND *mature* is nothing more than the intersection (AND) operation defined earlier. In the present case, the independent variables of the membership functions of inputs and outputs are different, so the result will be two-dimensional. For instance, Figs. 3.49(a) and (b) show the membership functions of *red* and *mature*, and Fig. 3.49(c) shows how they relate in two dimensions. To find the result of the AND operation between these two functions, recall from Eq. (3.8-5) that AND is defined as the minimum of the two membership functions; that is,

$$\mu_3(z, v) = \min\{\mu_{red}(z), \mu_{mat}(v)\} \tag{3.8-12}$$

a b
c d

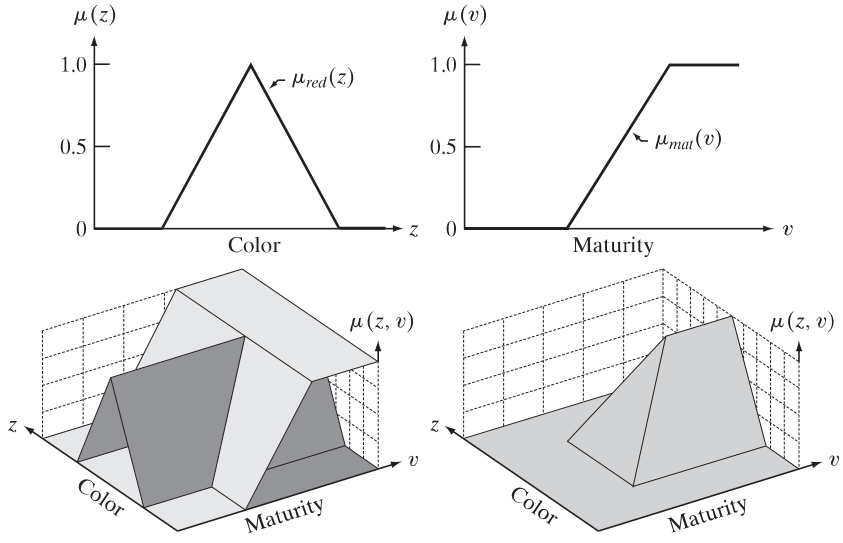


FIGURE 3.49 (a) Shape of the membership function associated with the color red, and (b) corresponding output membership function. These two functions are associated by rule R_3. (c) Combined representation of the two functions. The representation is 2-D because the independent variables in (a) and (b) are different. (d) The AND of (a) and (b), as defined in Eq. (3.8-5).

where 3 in the subscript denotes that this is the result of rule R_3 in the knowledge base. Figure 3.49(d) shows the result of the AND operation.[†]

Equation (3.8-12) is a general result involving two membership functions. In practice, we are interested in the output resulting from a *specific* input. Let z_0 denote a specific value of *red*. The degree of membership of the red color component in response to this input is simply a scalar value, $\mu_{red}(z_0)$. We find the output corresponding to rule R_3 and this specific input by performing the AND operation between $\mu_{red}(z_0)$ and the general result, $\mu_3(z, v)$, evaluated also at z_0. As noted before, the AND operation is implemented using the minimum operation:

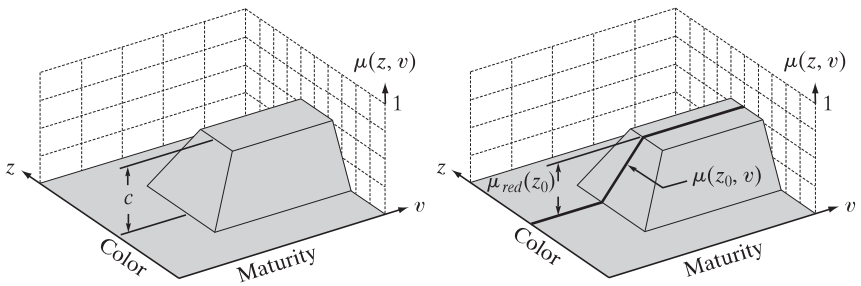
$$Q_3(v) = \min\{\mu_{red}(z_0), \mu_3(z_0, v)\} \tag{3.8-13}$$

where $Q_3(v)$ denotes the fuzzy output due to rule R_3 and a specific input. The only variable in Q_3 is the output variable, v, as expected.

To interpret Eq. (3.8-13) graphically, consider Fig. 3.49(d) again, which shows the general function $\mu_3(z, v)$. Performing the minimum operation of a positive constant, c, and this function would clip all values of $\mu_3(z, v)$ above that constant, as Fig. 3.50(a) shows. However, we are interested only in one value (z_0) along the color axis, so the relevant result is a cross section of the truncated function along the maturity axis, with the cross section placed at z_0, as Fig. 3.50(b) shows [because Fig. 3.50(a) corresponds to rule R_3, it follows that $c = \mu_{red}(z_0)$]. Equation (3.8-13) is the expression for this cross section.

Using the same line of reasoning, we obtain the fuzzy responses due to the other two rules and the specific input z_0, as follows:

$$Q_2(v) = \min\{\mu_{yellow}(z_0), \mu_2(z_0, v)\} \tag{3.8-14}$$



a b

FIGURE 3.50 (a) Result of computing the minimum of an arbitrary constant, c, and function $\mu_3(z, v)$ from Eq. (3.8-12). The minimum is equivalent to an AND operation. (b) Cross section (dark line) at a specific color, z_0.

[†] Note that Eq. (3.8-12) is formed from ordered pairs of values $\{\mu_{red}(z), \mu_{mat}(v)\}$, and recall that a set of ordered pairs is commonly called a *Cartesian product*, denoted by $X \times V$, where X is a set of values $\{\mu_{red}(z_1), \mu_{red}(z_2), \ldots, \mu_{red}(z_n)\}$ generated from $\mu_{red}(z)$ by varying z, and V is a similar set of n values generated from $\mu_{med}(v)$ by varying v. Thus, $X \times V = \{(\mu_{red}(z_1), \mu_{med}(v_1)), \ldots, (\mu_{red}(z_n), \mu_{med}(v_n))\}$, and we see from Fig. 3.49(d) that the AND operation involving two variables can be expresses as a mapping from $X \times V$ to the range $[0, 1]$, denoted as $X \times V \rightarrow [0, 1]$. Although we do not use this notation in the present discussion, we mention it here because you are likely to encounter it in the literature on fuzzy sets.

and

$$Q_1(v) = \min\{\mu_{green}(z_0), \mu_1(z_0, v)\} \tag{3.8-15}$$

Each of these equations is the output associated with a particular rule and a specific input. That is, they represent the result of the implication process mentioned a few paragraphs back. Keep in mind that each of these three responses is a fuzzy set, even though the input is a scalar value.

To obtain the overall response, we *aggregate* the individual responses. In the rule base given at the beginning of this section the three rules are associated by the OR operation. Thus, the complete (aggregated) fuzzy output is given by

$$Q = Q_1 \text{ OR } Q_2 \text{ OR } Q_3 \tag{3.8-16}$$

and we see that the overall response is the union of three individual fuzzy sets. Because OR is defined as a max operation, we can write this result as

$$Q(v) = \max_r\Big\{\min_s\{\mu_s(z_0), \mu_r(z_0, v)\}\Big\} \tag{3.8-17}$$

for $r = \{1, 2, 3\}$ and $s = \{green, yellow, red\}$. Although it was developed in the context of an example, this expression is perfectly general; to extend it to n rules, we simply let $r = \{1, 2, \ldots, n\}$; similarly, we can expand s to include any finite number of membership functions. Equations (3.8-16) and (3.8-17) say the same thing: The response, Q, of our fuzzy system is the union of the individual fuzzy sets resulting from each rule by the implication process.

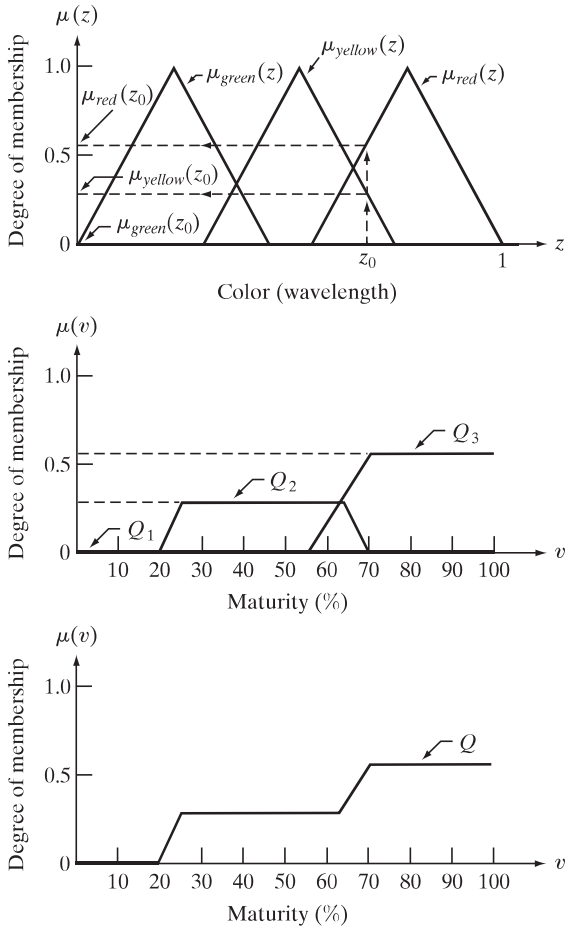
Figure 3.51 summarizes graphically the discussion up to this point. Figure 3.51(a) shows the three input membership functions evaluated at z_0, and Fig. 3.51(b) shows the outputs in response to input z_0. These fuzzy sets are the clipped cross sections discussed in connection with Fig. 3.50(b). Note that, numerically, Q_1 consists of all 0s because $\mu_{green}(z_0) = 0$; that is, Q_1 is *empty*, as defined in Section 3.8.2. Figure 3.51(c) shows the final result, Q, itself a fuzzy set formed from the union of Q_1, Q_2, and Q_3.

We have successfully obtained the complete output corresponding to a specific input, but we are still dealing with a fuzzy set. The last step is to obtain a crisp output, v_0, from fuzzy set Q using a process appropriately called *defuzzification*. There are a number of ways to defuzzify Q to obtain a crisp output. One of the approaches used most frequently is to compute the center of gravity of this set (the references cited at the end of this chapter discuss others). Thus, if $Q(v)$ from Eq. (3.8-17) can have K possible values, $Q(1), Q(2), \ldots Q(K)$, its center of gravity is given by

$$v_0 = \frac{\sum_{v=1}^{K} vQ(v)}{\sum_{v=1}^{K} Q(v)} \tag{3.8-18}$$

Evaluating this equation with the (discrete)[†] values of Q in Fig. 3.51(c) yields $v_0 = 72.3$, indicating that the given color z_0 implies a fruit maturity of approximately 72%.

[†] Fuzzy set Q in Fig. 3.51(c) is shown as a solid curve for clarity, but keep in mind that we are dealing with digital quantities in this book, so Q is a digital function.



a
b
c

FIGURE 3.51
(a) Membership functions with a specific color, z_0, selected. (b) Individual fuzzy sets obtained from Eqs. (3.8-13)–(3.8-15). (c) Final fuzzy set obtained by using Eq. (3.8-16) or (3.8-17).

Up to this point, we have considered IF-THEN rules whose antecedents have only one part, such as "IF the color is *red*." Rules containing more than one part must be combined to yield a *single* number that represents the entire antecedent for that rule. For example, suppose that we have the rule: IF the color is *red* OR the consistency is *soft,* THEN the fruit is *mature*. A membership function would have to be defined for the linguistic variable *soft*. Then, to obtain a single number for this rule that takes into account both parts of the antecedent, we first evaluate a given input color value of *red* using the *red* membership function and a given value of *consistency* using the *soft* membership function. Because the two parts are linked by OR, we use the maximum of the two resulting values.† This value is then used in the implication process to "clip" the *mature* output membership function, which is the function associated with this rule. The rest of the procedure is as before, as the following summary illustrates.

†Antecedents whose parts are connected by ANDs are similarly evaluated using the min operation.

Figure 3.52 shows the fruit example using two inputs: *color* and *consistency*. We can use this figure and the preceding material to summarize the principal steps followed in the application of rule-based fuzzy logic:

1. *Fuzzify the inputs:* For each scalar input, find the corresponding fuzzy values by mapping that input to the interval [0, 1], using the applicable membership functions in each rule, as the first two columns of Fig. 3.52 show.
2. *Perform any required fuzzy logical operations:* The outputs of all parts of an antecedent must be combined to yield a *single* value using the max or min operation, depending on whether the parts are connected by ORs or by ANDs. In Fig. 3.52, all the parts of the antecedents are connected by

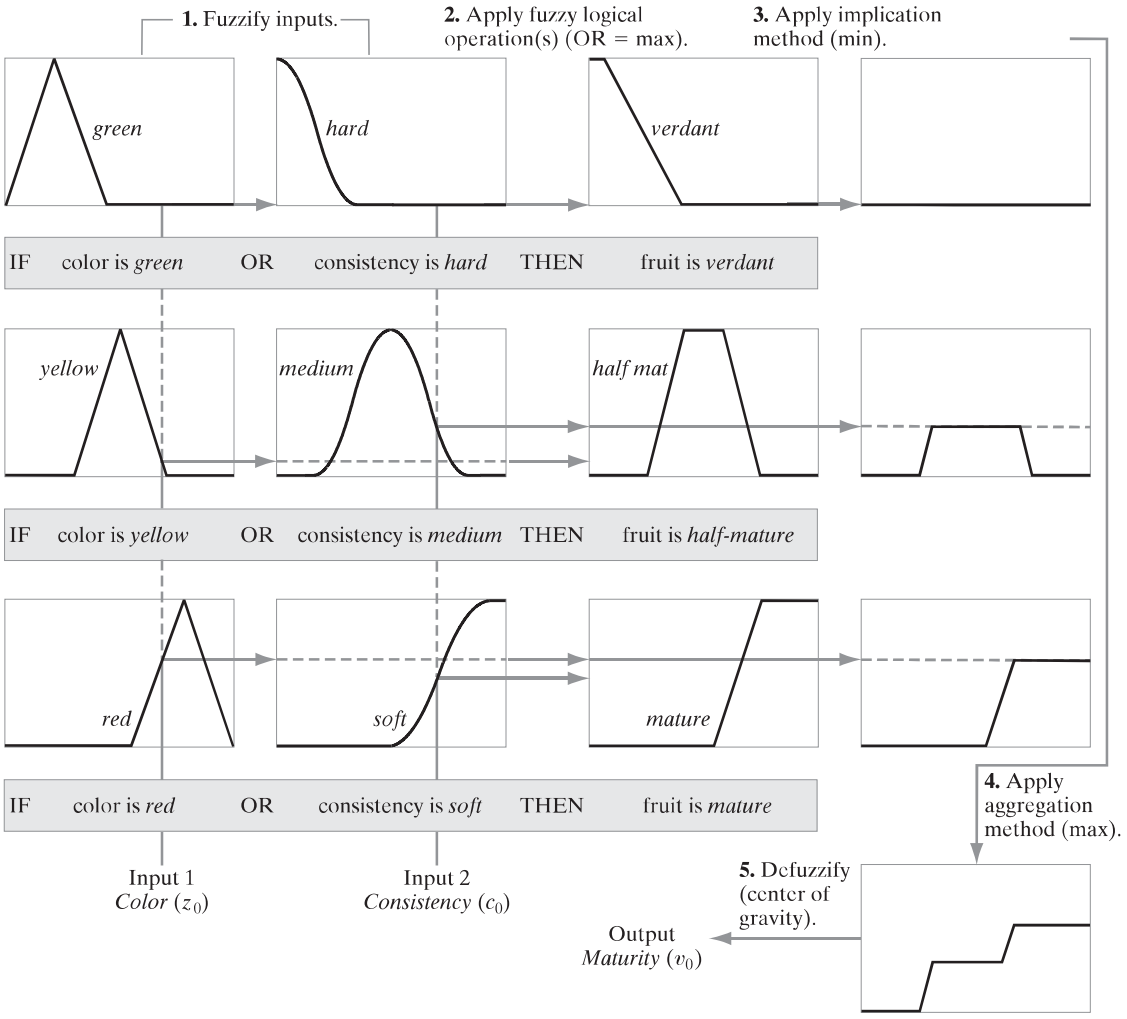


FIGURE 3.52 Example illustrating the five basic steps used typically to implement a fuzzy, rule-based system: (1) fuzzification, (2) logical operations (only OR was used in this example), (3) implication, (4) aggregation, and (5) defuzzification.

ORs, so the max operation is used throughout. The number of parts of an antecedent and the type of logic operator used to connect them can be different from rule to rule.

3. *Apply an implication method:* The single output of the antecedent of each rule is used to provide the output corresponding to that rule. We use AND for implication, which is defined as the min operation. This clips the corresponding output membership function at the value provided by the antecedent, as the third and fourth columns in Fig. 3.52 show.
4. *Apply an aggregation method to the fuzzy sets from step 3:* As the last column in Fig. 3.52 shows, the output of each rule is a fuzzy set. These must be combined to yield a single output fuzzy set. The approach used here is to OR the individual outputs, so the max operation is employed.
5. *Defuzzify the final output fuzzy set:* In this final step, we obtain a crisp, scalar output. This is achieved by computing the center of gravity of the aggregated fuzzy set from step 4.

When the number of variables is large, it is common practice to use the shorthand notation (variable, fuzzy set) to pair a variable with its corresponding membership function. For example, the rule IF the color is *green* THEN the fruit is *verdant* would be written as IF (z, *green*) THEN (v, *verdant*) where, as before, variables z and v represent color and degree of maturity, respectively, while *green* and *verdant* are the two fuzzy sets defined by the membership functions $\mu_{green}(z)$ and $\mu_{verd}(v)$, respectively.

In general, when dealing with M IF-THEN rules, N input variables, $z_1, z_2, \ldots z_N$, and one output variable, v, the type of fuzzy rule formulation used most frequently in image processing has the form

$$\begin{aligned}
&\text{IF } (z_1, A_{11}) \text{ AND } (z_2, A_{12}) \text{ AND} \ldots \text{AND } (z_N, A_{1N}) \text{ THEN } (v, B_1)\\
&\text{IF } (z_1, A_{21}) \text{ AND } (z_2, A_{22}) \text{ AND} \ldots \text{AND } (z_N, A_{2N}) \text{ THEN } (v, B_2)\\
&\ldots\ldots\\
&\text{IF } (z_1, A_{M1}) \text{ AND } (z_2, A_{M2}) \text{ AND} \ldots \text{AND } (z_N, A_{MN}) \text{ THEN } (v, B_M)\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{ELSE } (v, B_E)
\end{aligned} \tag{3.8-19}$$

where A_{ij} is the fuzzy set associated with the ith rule and the jth input variable, B_i is the fuzzy set associated with the output of the ith rule, and we have assumed that the components of the rule antecedents are linked by ANDs. Note that we have introduced an ELSE rule, with associated fuzzy set B_E. This rule is executed when none of the preceding rules is completely satisfied; its output is explained below.

As indicated earlier, all the elements of the antecedent of each rule must be evaluated to yield a single scalar value. In Fig. 3.52, we used the max operation because the rules were based on fuzzy ORs. The formulation in Eq. (3.8-19) uses ANDs, so we have to use the min operator. Evaluating the antecedents of the ith rule in Eq. (3.8-19) produces a scalar output, λ_i, given by

$$\lambda_i = \min\left\{\mu_{A_{ij}}(z_j); \quad j = 1, 2, \ldots, M\right\} \tag{3.8-20}$$

The use of OR or AND in the rule set depends on how the rules are stated, which in turn depends on the problem at hand. We used ORs in Fig. 3.52 and ANDs in Eq. (3.8-19) to give you familiarity with both formulations.

for $i = 1, 2, \ldots, M$, where $\mu_{A_{ij}}(z_j)$ is the membership function of fuzzy set A_{ij} evaluated at the value of the jth input. Often, λ_i is called the *strength level* (or *firing level*) of the ith rule. With reference to the preceding discussion, λ_i is simply the value used to clip the output function of the ith rule.

The ELSE rule is executed when the conditions of the THEN rules are weakly satisfied (we give a detailed example of how ELSE rules are used in Section 3.8.5). Its response should be strong when all the others are weak. In a sense, one can view an ELSE rule as performing a NOT operation on the results of the other rules. We know from Section 3.8.2 that $\mu_{\text{NOT}(A)} = \mu_{\bar{A}}(z) = 1 - \mu_A(z)$. Then, using this idea in combining (ANDing) all the levels of the THEN rules gives the following strength level for the ELSE rule:

$$\lambda_E = \min\{1 - \lambda_i; \quad i = 1, 2, \ldots, M\} \tag{3.8-21}$$

We see that if all the THEN rules fire at "full strength" (all their responses are 1), then the response of the ELSE rule is 0, as expected. As the responses of the THEN rules weaken, the strength of the ELSE rule increases. This is the fuzzy counterpart of the familiar IF-THEN-ELSE rules used in software programming.

When dealing with ORs in the antecedents, we simply replace the ANDs in Eq. (3.8-19) by ORs and the min in Eq. (3.8-20) by a max; Eq. (3.8-21) does not change. Although one could formulate more complex antecedents and consequents than the ones discussed here, the formulations we have developed using only ANDs or ORs are quite general and are used in a broad spectrum of image processing applications. The references at the end of this chapter contain additional (but less used) definitions of fuzzy logical operators, and discuss other methods for implication (including multiple outputs) and defuzzification. The introduction presented in this section is fundamental and serves as a solid base for more advanced reading on this topic. In the next two sections, we show how to apply fuzzy concepts to image processing.

3.8.4 Using Fuzzy Sets for Intensity Transformations

Consider the general problem of contrast enhancement, one of the principal applications of intensity transformations. We can state the process of enhancing the contrast of a gray-scale image using the following rules:

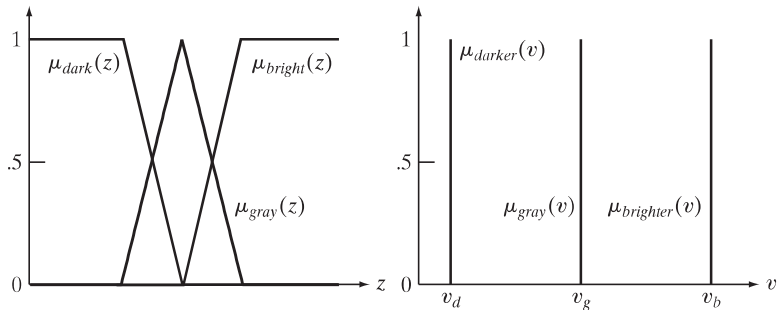
IF a pixel is dark, THEN make it *darker*.

IF a pixel is gray, THEN make it gray.

IF a pixel is bright, THEN make it brighter.

Keeping in mind that these are fuzzy terms, we can express the concepts of *dark*, *gray*, and *bright* by the membership functions in Fig. 3.53(a).

In terms of the output, we can consider *darker* as being degrees of a dark intensity value (100% black being the limiting shade of dark), *brighter*, as being degrees of a bright shade (100% white being the limiting value), and *gray* as being degrees of an intensity in the middle of the gray scale. What we mean by



a b

FIGURE 3.53 (a) Input and (b) output membership functions for fuzzy, rule-based contrast enhancement.

"degrees" here is the amount of one specific intensity. For example, 80% black is a very dark gray. When interpreted as *constant* intensities whose strength is modified, the output membership functions are *singletons* (membership functions that are *constant*), as Fig. 3.53(b) shows. The various degrees of an intensity in the range $[0, 1]$ occur when the singletons are clipped by the strength of the response from their corresponding rules, as in the fourth column of Fig. 3.52 (but keep in mind that we are working here with only one input, not two, as in the figure). Because we are dealing with constants in the output membership functions, it follows from Eq. (3.8-18) that the output, v_0, to any input, z_0, is given by

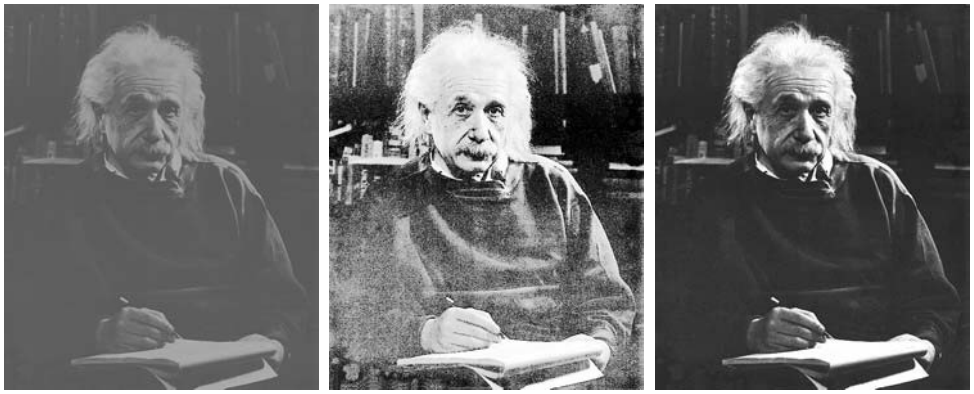
$$v_0 = \frac{\mu_{dark}(z_0) \times v_d + \mu_{gray}(z_0) \times v_g + \mu_{bright}(z_0) \times v_b}{\mu_{dark}(z_0) + \mu_{gray}(z_0) + \mu_{bright}(z_0)} \quad (3.8\text{-}22)$$

The summations in the numerator and denominator in this expressions are simpler than in Eq. (3.8-18) because the output membership functions are constants modified (clipped) by the fuzzified values.

Fuzzy image processing is computationally intensive because the entire process of fuzzification, processing the antecedents of all rules, implication, aggregation, and defuzzification must be applied to *every* pixel in the input image. Thus, using singletons as in Eq. (3.8-22) significantly reduces computational requirements by simplifying implication, aggregation, and defuzzification. These savings can be significant in applications where processing speed is an important requirement.

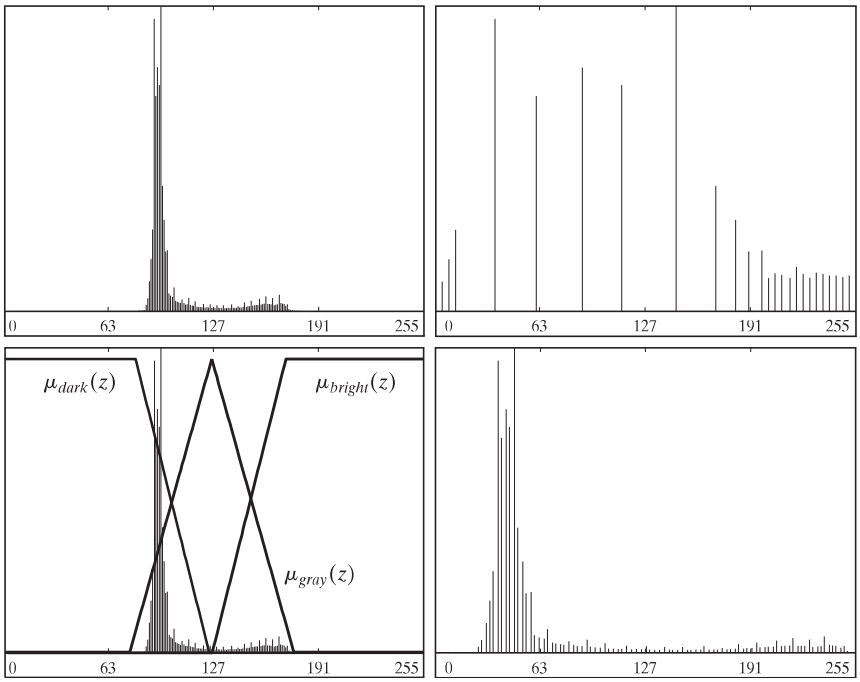
EXAMPLE 3.19: Illustration of image enhancement using fuzzy, rule-based contrast modification.

■ Figure 3.54(a) shows an image whose intensities span a narrow range of the gray scale [see the image histogram in Fig. 3.55(a)], thus giving the image an appearance of low contrast. As a basis for comparison, Fig. 3.54(b) is the result of histogram equalization. As the histogram of this result shows [Fig. 3.55(b)], expanding the entire gray scale does increase contrast, but introduces intensities in the high and low end that give the image an "overexposed" appearance. For example, the details in Professor Einstein's forehead and hair are mostly lost. Figure 3.54(c) shows the result of using the rule-based contrast modification approach discussed in the preceding paragraphs. Figure 3.55(c) shows the input membership functions used, superimposed on the histogram of the original image. The output singletons were selected at $v_d = 0$ (black), $v_g = 127$ (mid gray), and $v_b = 255$ (white).



a b c

FIGURE 3.54 (a) Low-contrast image. (b) Result of histogram equalization. (c) Result of using fuzzy, rule-based contrast enhancement.



a b
c d

FIGURE 3.55 (a) and (b) Histograms of Figs. 3.54(a) and (b). (c) Input membership functions superimposed on (a). (d) Histogram of Fig. 3.54(c).

Comparing Figs. 3.54(b) and 3.54(c), we see in the latter a considerable improvement in tonality. Note, for example, the level of detail in the forehead and hair, as compared to the same regions in Fig. 3.54(b). The reason for the improvement can be explained easily by studying the histogram of Fig. 3.54(c), shown in Fig. 3.55(d). Unlike the histogram of the equalized image, this histogram has kept the same basic characteristics of the histogram of the original image. However, it is quite evident that the dark levels (talk peaks in the low end of the histogram) were moved left, thus darkening the levels. The opposite was true for bright levels. The mid grays were spread slightly, but much less than in histogram equalization.

The price of this improvement in performance is considerably more processing complexity. A practical approach to follow when processing speed and image throughput are important considerations is to use fuzzy techniques to determine what the histograms of well-balanced images should look like. Then, faster techniques, such as histogram specification, can be used to achieve similar results by mapping the histograms of the input images to one or more of the "ideal" histograms determined using a fuzzy approach. ■

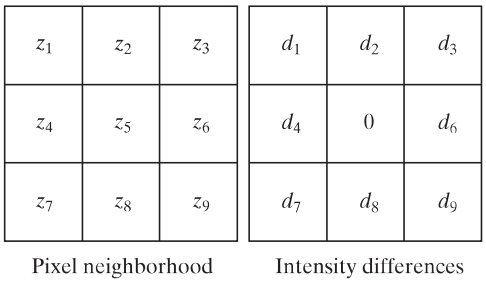
3.8.5 Using Fuzzy Sets for Spatial Filtering

When applying fuzzy sets to spatial filtering, the basic approach is to define neighborhood properties that "capture" the essence of what the filters are supposed to detect. For example, consider the problem of detecting boundaries between regions in an image. This is important in numerous applications of image processing, such as sharpening, as discussed earlier in this section, and in image segmentation, as discussed in Chapter 10.

We can develop a boundary extraction algorithm based on a simple fuzzy concept: *If a pixel belongs to a uniform region, then make it white; else make it black*, where, *black* and *white* are fuzzy sets. To express the concept of a "uniform region" in fuzzy terms, we can consider the intensity differences between the pixel at the center of a neighborhood and its neighbors. For the 3×3 neighborhood in Fig. 3.56(a), the differences between the center pixel (labeled z_5) and each of the neighbors forms the subimage of size 3×3 in Fig. 3.56(b), where d_i denotes the intensity difference between the ith neighbor and the center point (i.e., $d_i = z_i - z_5$, where the zs are intensity values). A simple set of four IF-THEN rules and one ELSE rule implements the essence of the fuzzy concept mentioned at the beginning of this paragraph:

IF d_2 is *zero* AND d_6 is *zero* THEN z_5 is *white*
IF d_6 is *zero* AND d_8 is *zero* THEN z_5 is *white*
IF d_8 is *zero* AND d_4 is *zero* THEN z_5 is *white*
IF d_4 is *zero* AND d_2 is *zero* THEN z_5 is *white*
ELSE z_5 is *black*

We used only the intensity differences between the 4-neighbors and the center point to simplify the example. Using the 8-neighbors would be a direct extension of the approach shown here.



a b

FIGURE 3.56 (a) A 3×3 pixel neighborhood, and (b) corresponding intensity differences between the center pixels and its neighbors. Only d_2, d_4, d_6, and d_8 were used in the present application to simplify the discussion.

where *zero* is a fuzzy set also. The consequent of each rule defines the values to which the intensity of the center pixel (z_5) is mapped. That is, the statement "THEN z_5 is white" means that the intensity of the pixel located at the center of the mask is mapped to white. These rules simply state that the center pixel is considered to be part of a uniform region if the intensity differences just mentioned are zero (in a fuzzy sense); otherwise it is considered a boundary pixel.

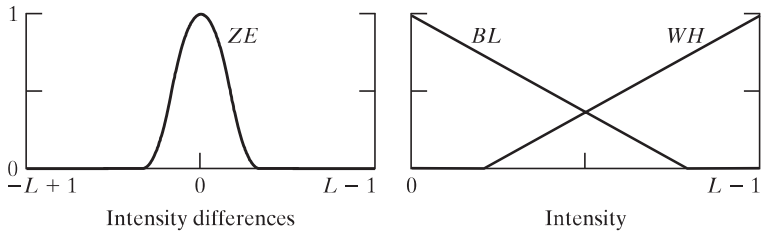
Figure 3.57 shows possible membership functions for the fuzzy sets *zero*, *black*, and *white*, respectively, where we used ZE, BL, and WH to simplify notation. Note that the range of the independent variable of the fuzzy set ZE for an image with L possible intensity levels is $[-L + 1, L - 1]$ because intensity differences can range between $-(L - 1)$ and $(L - 1)$. On the other hand, the range of the output intensities is $[0, L - 1]$, as in the original image. Figure 3.58 shows graphically the rules stated above, where the box labeled z_5 indicates that the intensity of the center pixel is mapped to the output value WH or BL.

EXAMPLE 3.20: Illustration of boundary enhancement using fuzzy, rule-based spatial filtering.

■ Figure 3.59(a) shows a 512×512 CT scan of a human head, and Fig. 3.59(b) is the result of using the fuzzy spatial filtering approach just discussed. Note the effectiveness of the method in extracting the boundaries between regions, including the contour of the brain (inner gray region). The constant regions in the image appear as gray because when the intensity differences discussed earlier are near zero, the THEN rules have a strong response. These responses in turn clip function WH. The output (the center of gravity of the clipped triangular regions) is a constant between $(L - 1)/2$ and $(L - 1)$, thus producing the grayish tone seen in the image. The contrast of this image can be improved significantly by expanding the

a b

FIGURE 3.57 (a) Membership function of the fuzzy set *zero*. (b) Membership functions of the fuzzy sets *black* and *white*.



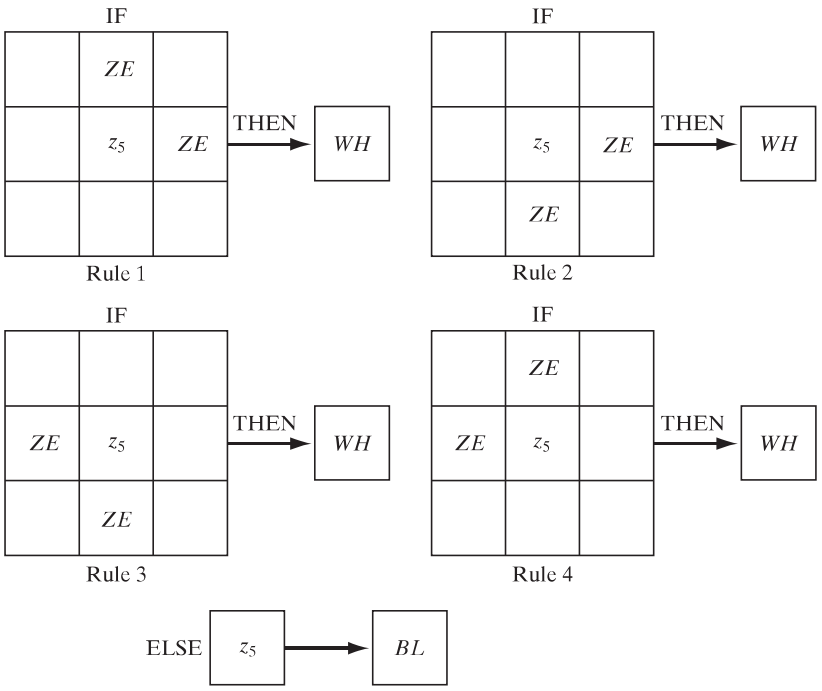
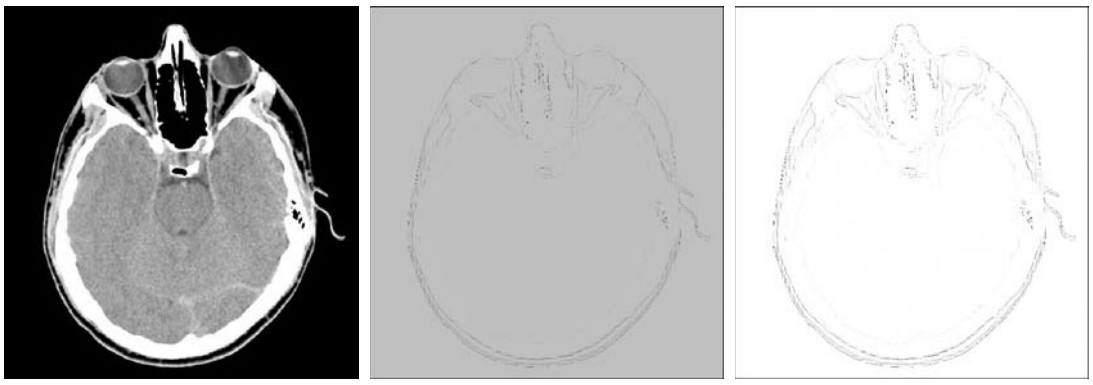


FIGURE 3.58 Fuzzy rules for boundary detection.

gray scale. For example, Fig. 3.59(c) was obtained by performing the intensity scaling defined in Eqs. (2.6-10) and (2.6-11), with $K = L - 1$. The net result is that intensity values in Fig. 3.59(c) span the full gray scale from 0 to $(L - 1)$. ■



a b c

FIGURE 3.59 (a) CT scan of a human head. (b) Result of fuzzy spatial filtering using the membership functions in Fig. 3.57 and the rules in Fig. 3.58. (c) Result after intensity scaling. The thin black picture borders in (b) and (c) were added for clarity; they are not part of the data. (Original image courtesy of Dr. David R. Pickens, Vanderbilt University.)

Summary

The material you have just learned is representative of current techniques used for intensity transformations and spatial filtering. The topics included in this chapter were selected for their value as fundamental material that would serve as a foundation in an evolving field. Although most of the examples used in this chapter were related to image enhancement, the techniques presented are perfectly general, and you will encounter them again throughout the remaining chapters in contexts totally unrelated to enhancement. In the following chapter, we look again at filtering, but using concepts from the frequency domain. As you will see, there is a one-to-one correspondence between the linear spatial filters studied here and frequency domain filters.

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 further reading on exact histogram specification see Coltuc, Bolon, and Chassery [2006]. 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 processing. Kim et al. [1997] present an interesting approach combining the gradient with local statistics for image enhancement.

For additional reading on linear spatial filters and their implementation, see Umbaugh [2005], 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 the *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).

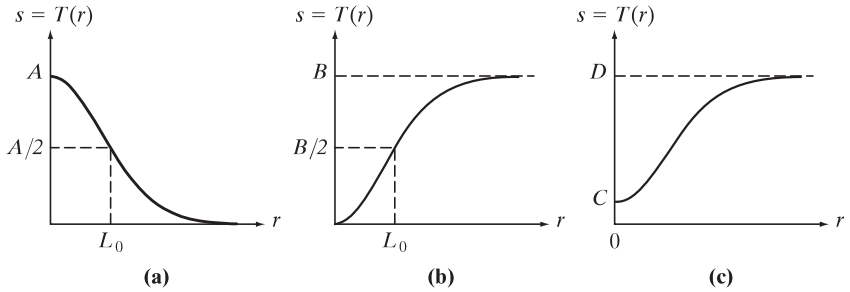
Fundamental references for Section 3.8 are three papers on fuzzy logic by L. A. Zadeh (Zadeh [1965, 1973, 1976]). These papers are well written and worth reading in detail, as they established the foundation for fuzzy logic and some of its applications. An overview of a broad range of applications of fuzzy logic in image processing can be found in the book by Kerre and Nachtegael [2000]. The example in Section 3.8.4 is based on a similar application described by Tizhoosh [2000]. The example in Section 3.8.5 is basically from Russo and Ramponi [1994]. For additional examples of applications of fuzzy sets to intensity transformations and image filtering, see Patrascu [2004] and Nie and Barner [2006], respectively. The preceding range of references from 1965 through 2006 is a good starting point for more detailed study of the many ways in which fuzzy sets can be used in image processing. Software implementation of most of the methods discussed in this chapter can be found in Gonzalez, Woods, and Eddins [2004].

Problems

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** Give a single intensity transformation function for spreading the intensities of an image so the lowest intensity is 0 and the highest is $L - 1$.

3.2 Exponentials of the form $e^{-\alpha r^2}$, with α a positive constant, are useful for constructing smooth intensity 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.)



3.3 ★**(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 intensity 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 intensity 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 $m = 128$. Let C denote the smallest positive number representable in the computer you are using.

3.4 Propose a set of intensity-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 8th bit plane in an 8-bit image.)

3.5 ★**(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?

★**3.6** Explain why the discrete histogram equalization technique does not, in general, yield a flat histogram.

3.7 Suppose that a digital image is subjected to histogram equalization. Show that a second pass of histogram equalization (on the histogram-equalized image) will produce exactly the same result as the first pass.

3.8 In some applications it is useful to model the histogram of input images as Gaussian probability density functions of the form

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

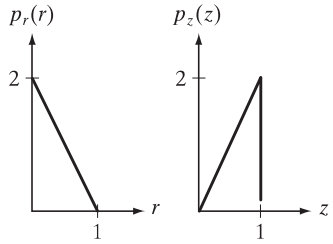
where m and σ are the mean and standard deviation of the Gaussian PDF. The approach is to let m and σ be measures of average intensity and contrast of a given image. What is the transformation function you would use for histogram equalization?

★**3.9** 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 condition (a′).

3.10 **(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 that the inverse discrete transformation in Eq. (3.3-9) satisfies conditions (a′) and (b) in Section 3.3.1 only if none of the intensity levels $r_k, k = 0, 1, \ldots, L-1$, are missing.

3.11 An image with intensities in the range $[0, 1]$ has the PDF $p_r(r)$ shown in the following diagram. It is desired to transform the intensity 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.



★**3.12** Propose a method for updating the local histogram for use in the local enhancement technique discussed in Section 3.3.3.

3.13 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) $f(x, y) \times g(x, y)$

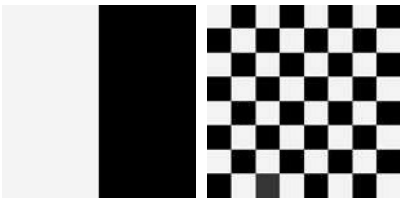
(d) $f(x, y) \div g(x, y)$

in terms of h_f and h_g. Explain how to obtain the histogram in each case.

3.14 The images shown on the next page are quite different, but their histograms are the same. Suppose that each image is blurred with a 3×3 averaging mask.

(a) Would the histograms of the blurred images still be equal? Explain.

(b) If your answer is no, sketch the two histograms.



3.15 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 3.4). A lowpass filter can be implemented by setting all coefficients to 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 wide enough to allow you to ignore border effects in your analysis.

3.16 ★ **(a)** Suppose that you filter an image, $f(x, y)$, with a spatial filter mask, $w(x, y)$, using convolution, as defined in Eq. (3.4-2), where the mask is smaller than the image in both spatial directions. Show the important property that, if the coefficients of the mask sum to zero, then the sum of all the elements in the resulting convolution array (filtered image) will be zero also (you may ignore computational inaccuracies). Also, you may assume that the border of the image has been padded with the appropriate number of zeros.

(b) Would the result to (a) be the same if the filtering is implemented using correlation, as defined in Eq. (3.4-1)?

3.17 Discuss the limiting effect of repeatedly applying a 3×3 lowpass spatial filter to a digital image. You may ignore border effects.

3.18 ★ **(a)** It was stated in Section 3.5.2 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)?

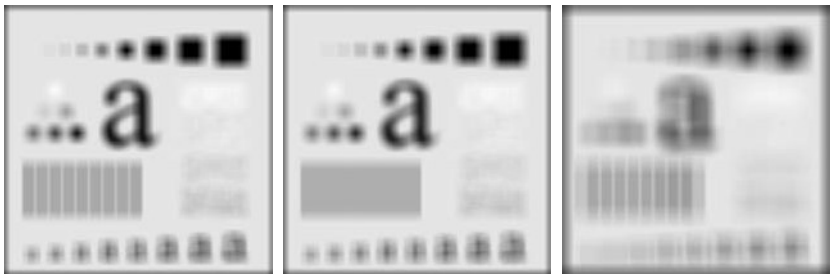
★3.19 **(a)** Develop a procedure for computing the median of an $n \times n$ neighborhood.

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

3.20 **(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 1s on a background of 0s. 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?

★3.21 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 the reason for this.



(a) **(b)** **(c)**

3.22 Consider an application such as the one shown in Fig. 3.34, in which it is desired to eliminate objects smaller than those enclosed by a square of size $q \times q$ pixels. Suppose that we want to reduce the average intensity of those objects to one-tenth of their original average value. In this way, those objects will be closer to the intensity 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 intensity in only one pass of the mask over the image.

3.23 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?

★**3.24** Show that the Laplacian defined in Eq. (3.6-3) is isotropic (invariant to rotation). You will need the following equations relating coordinates for 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.

★**3.25** You saw in Fig. 3.38 that the Laplacian with a -8 in the center yields sharper results than the one with a -4 in the center. Explain the reason in detail.

3.26 With reference to Problem 3.25,

(a) Would using a larger "Laplacian-like" mask, say, of size 5×5 with a -24 in the center, yield an even sharper result? Explain in detail.

(b) How does this type of filtering behave as a function of mask size?

3.27 Give a 3×3 mask for performing unsharp masking in a single pass through an image. Assume that the average image is obtained using the filter in Fig. 3.32(a).

★**3.28** Show that subtracting the Laplacian from an image is proportional to unsharp masking. Use the definition for the Laplacian given in Eq. (3.6-6).

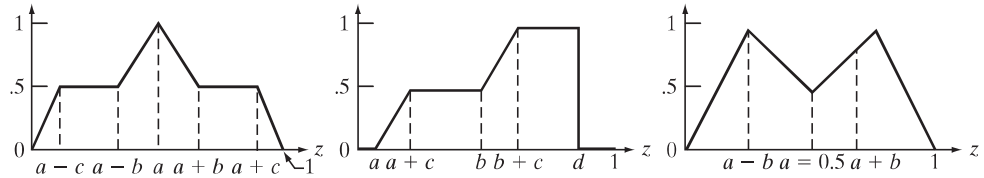
3.29 **(a)** Show that the magnitude of the gradient given in Eq. (3.6-11) is an isotropic operation. (See Problem 3.24.)

(b) Show that the isotropic property is lost in general if the gradient is computed using Eq. (3.6-12).

3.30 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 image processing techniques to post-process, 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.

3.31 Show that the crossover point in Fig. 3.46(d) is given by $b = (a + c)/2$.

3.32 Use the fuzzy set definitions in Section 3.8.2 and the basic membership functions in Fig. 3.46 to form the membership functions shown below.



★ **(a)** **(b)** **(c)**

★**3.33** What would be the effect of increasing the neighborhood size in the fuzzy filtering approach discussed in Section 3.8.5? Explain the reasoning for your answer (you may use an example to support your answer).

3.34 Design a fuzzy, rule-based system for reducing the effects of impulse noise on a noisy image with intensity values in the interval $[0, L-1]$. As in Section 3.8.5, use only the differences d_2, d_4, d_6, and d_8 in a 3×3 neighborhood in order to simplify the problem. Let z_5 denote the intensity at the center of the neighborhood, anywhere in the image. The corresponding output intensity values should be $z'_5 = z_5 + v$, where v is the output of your fuzzy system. That is, the output of your fuzzy system is a correction factor used to *reduce* the effect of a noise spike that may be present at the center of the 3×3 neighborhood. Assume that the noise spikes occur sufficiently apart so that you need not be concerned with multiple noise spikes being present in the same neighborhood. The spikes can be dark or light. Use triangular membership functions throughout.

★**(a)** Give a fuzzy statement for this problem.

★**(b)** Specify the IF-THEN and ELSE rules.

(c) Specify the membership functions graphically, as in Fig. 3.57.

(d) Show a graphical representation of the rule set, as in Fig. 3.58.

(e) Give a summary diagram of your fuzzy system similar to the one in Fig. 3.52.