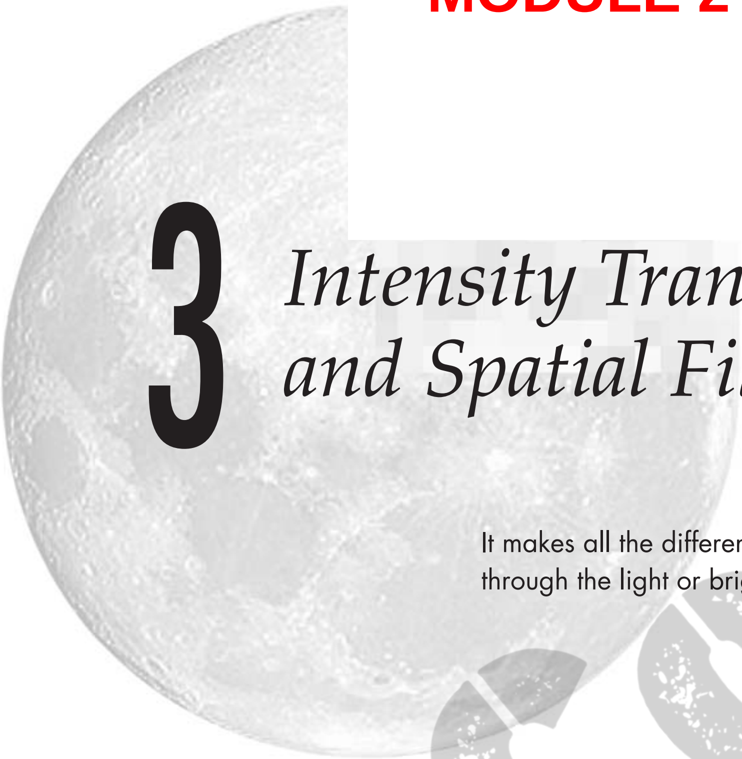
MODULE 2



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.

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

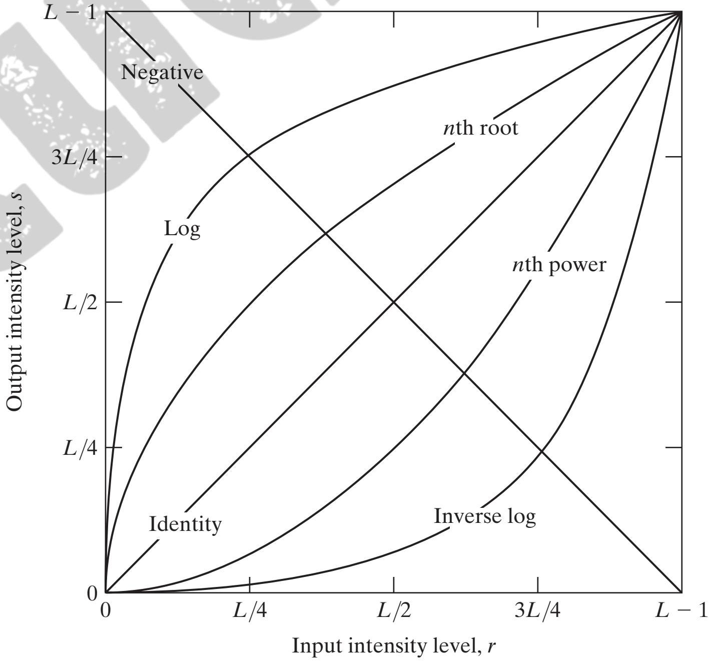
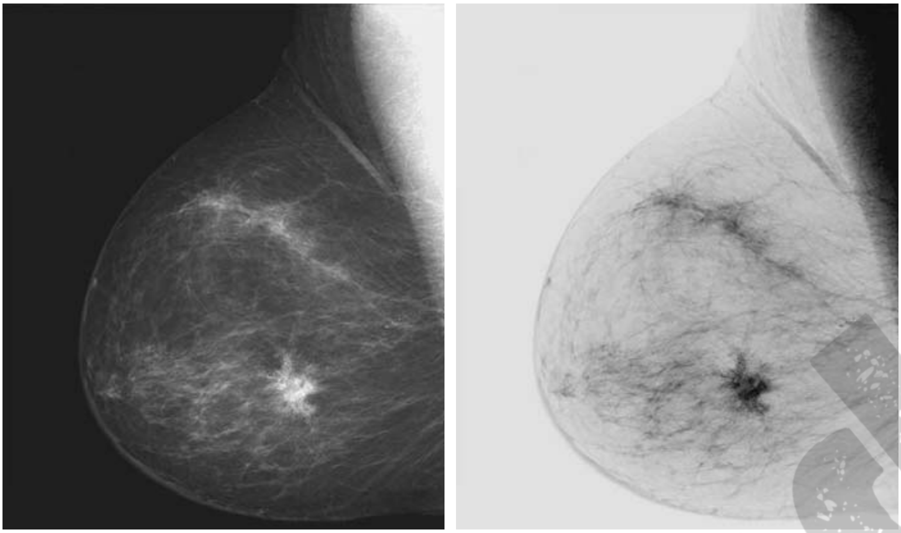


FIGURE 3.3 Some basic intensity transformation functions. All curves were scaled to fit in the range shown.



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.

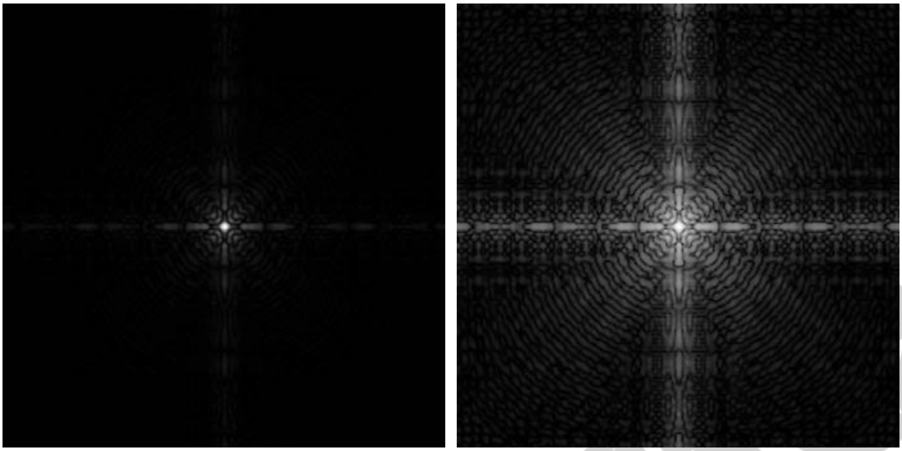
3.2.2 Log Transformations

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

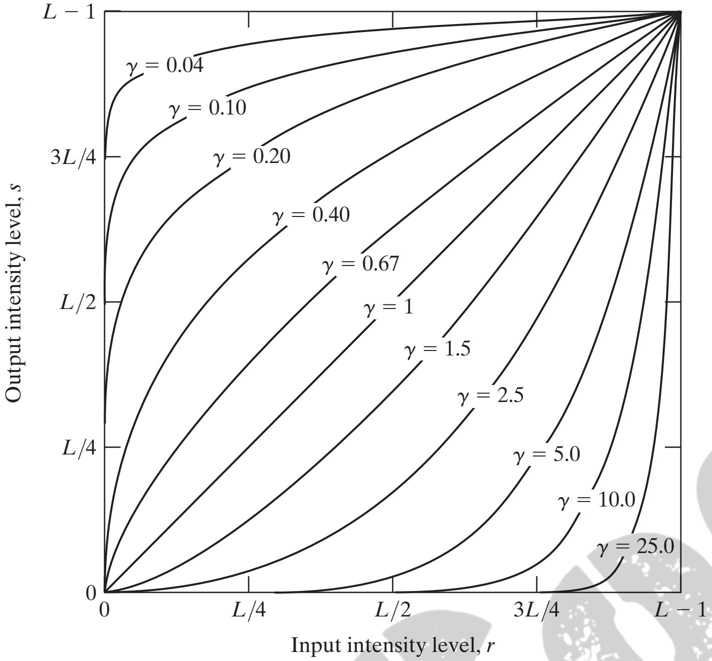
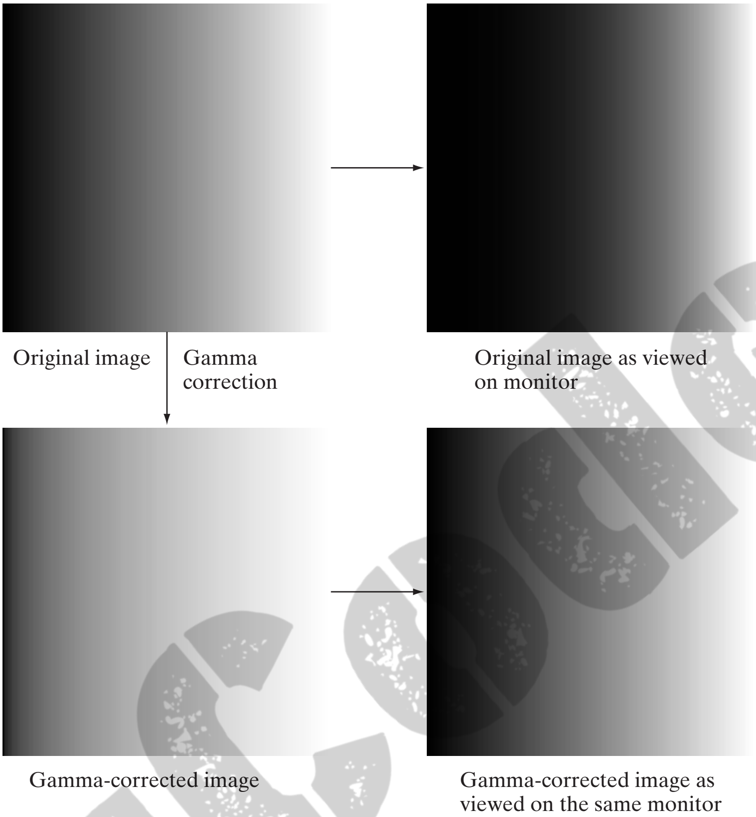


FIGURE 3.6 Plots of the equation $s = cr^{\gamma}$ for various values of γ ($c = 1$ in all cases). All curves were scaled to fit in the range shown.

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]).


Original image
Gamma correction
Original image as viewed on monitor
Gamma-corrected image
Gamma-corrected image as viewed on the same monitor

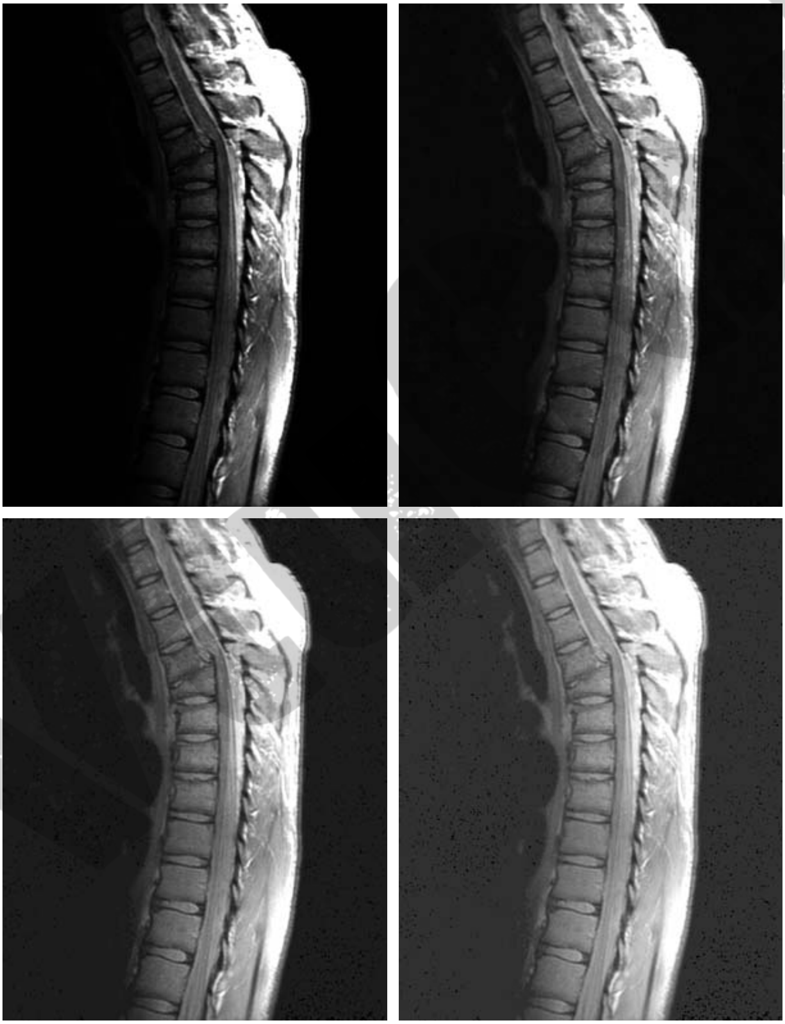
a b
c d

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.

EXAMPLE 3.1: Contrast enhancement using power-law transformations.

■ In addition to gamma correction, power-law transformations are useful for general-purpose contrast manipulation. Figure 3.8(a) shows a magnetic resonance 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



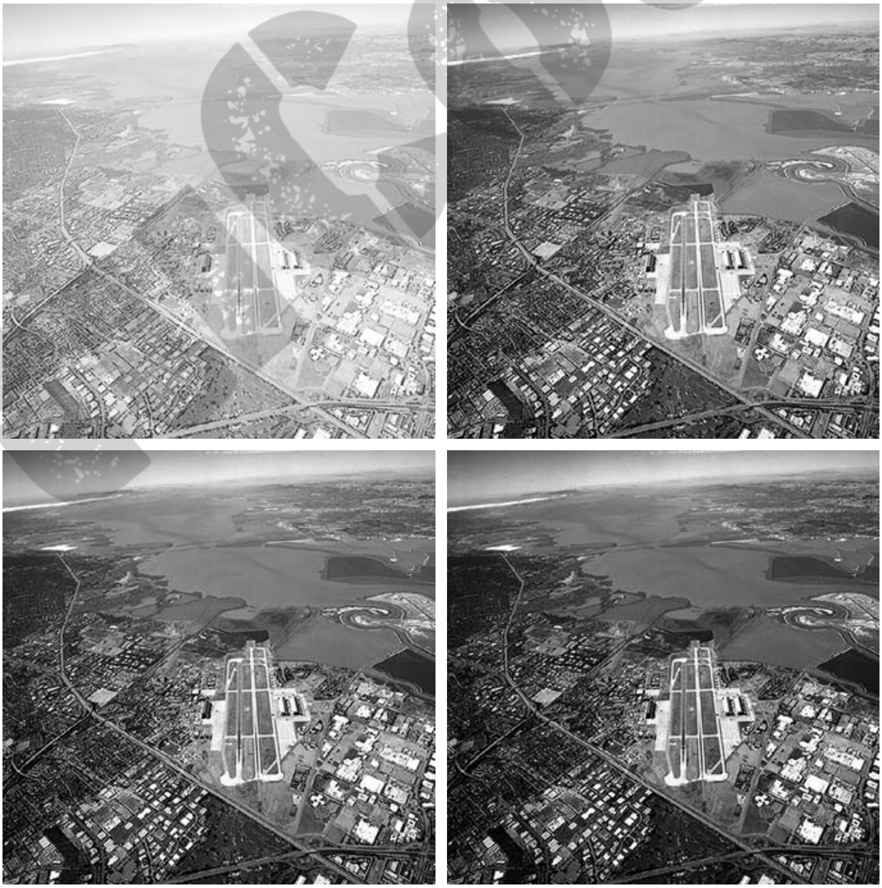
a b
c d

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

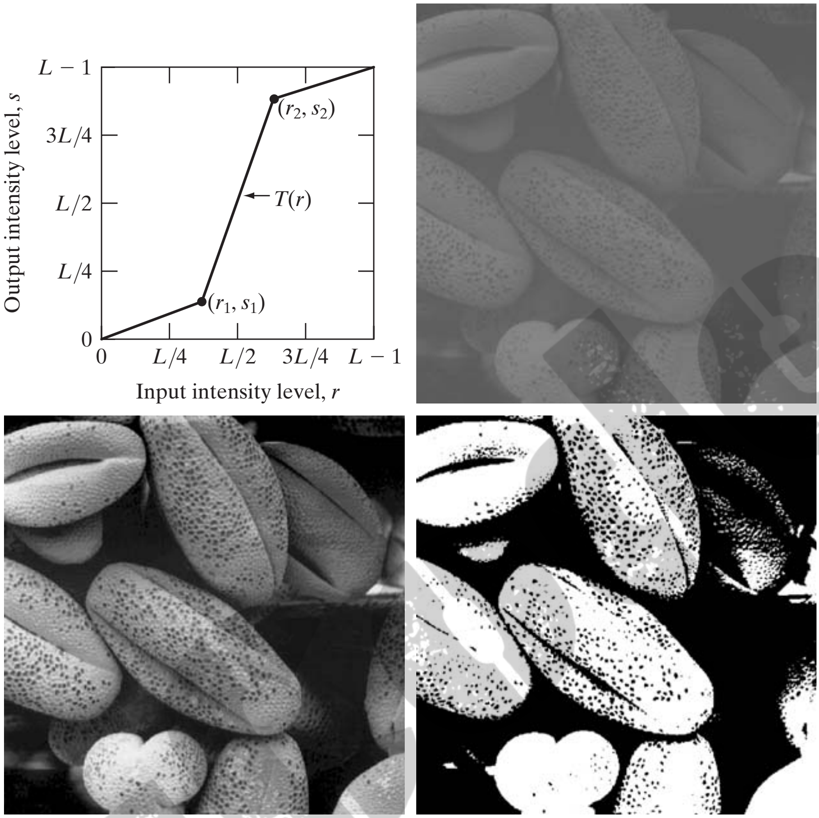
One of the simplest piecewise linear functions is a contrast-stretching transformation. Low-contrast images can result from poor illumination, lack of dynamic range in the imaging sensor, or even the wrong setting of a lens aperture during image acquisition. *Contrast stretching* is a process that expands 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

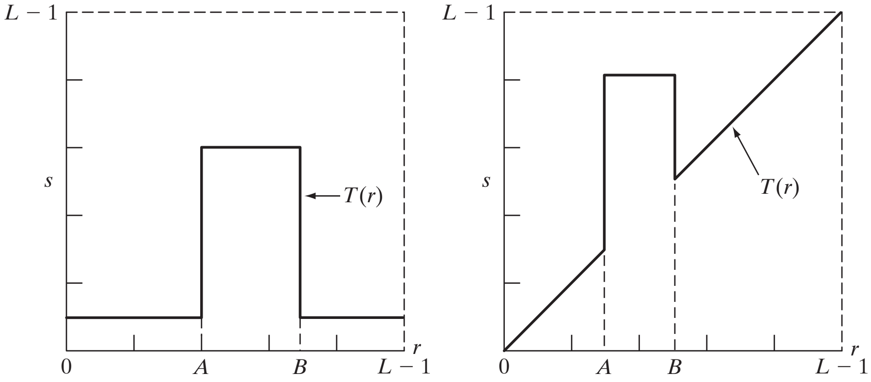
Highlighting a specific range of intensities in an image often is of interest. Applications include enhancing features such as masses of water in satellite imagery and enhancing flaws in X-ray images. The process, often called *intensity-level*



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.

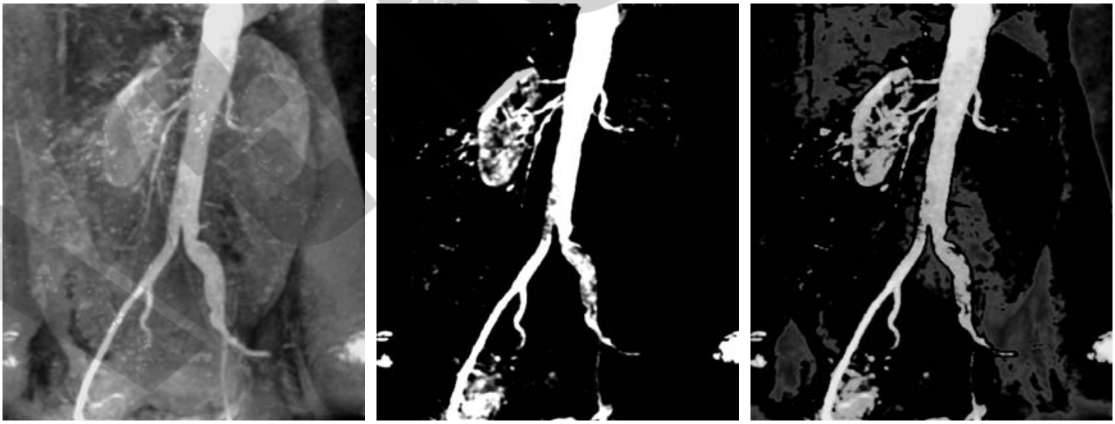
EXAMPLE 3.3: Intensity-level slicing.

■ 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.)

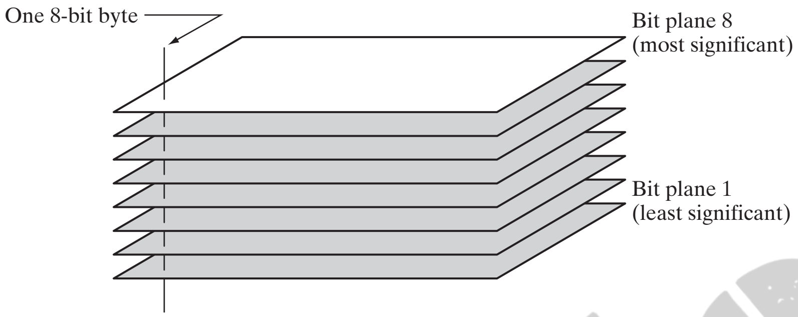
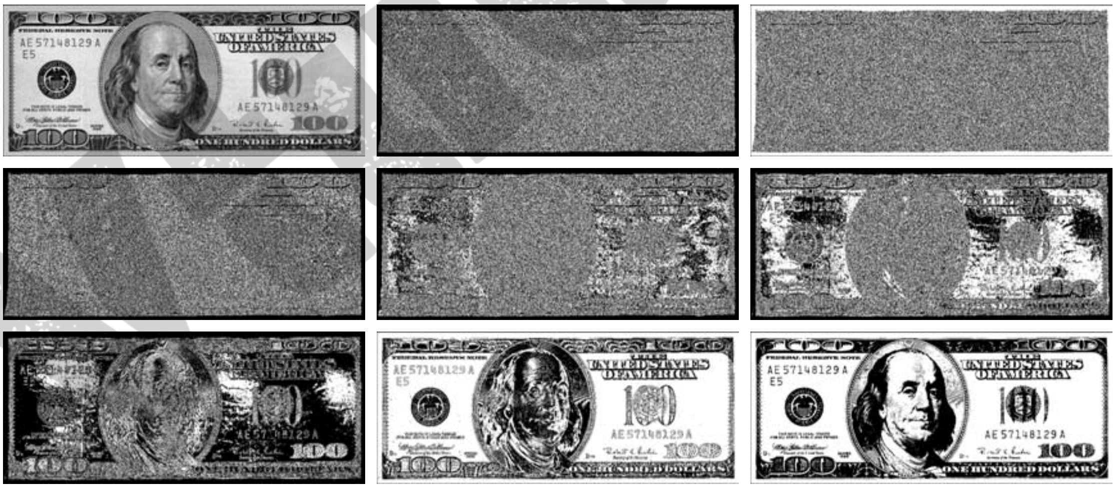


FIGURE 3.13 Bit-plane representation of an 8-bit image.

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



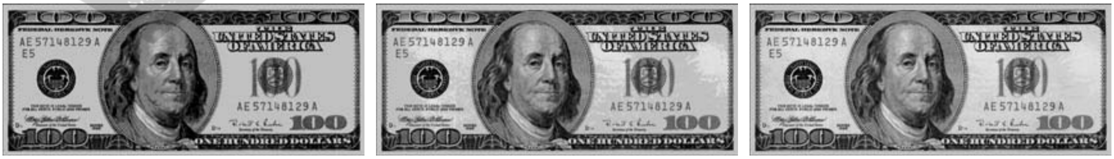
a b c
d e f
g h i

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.

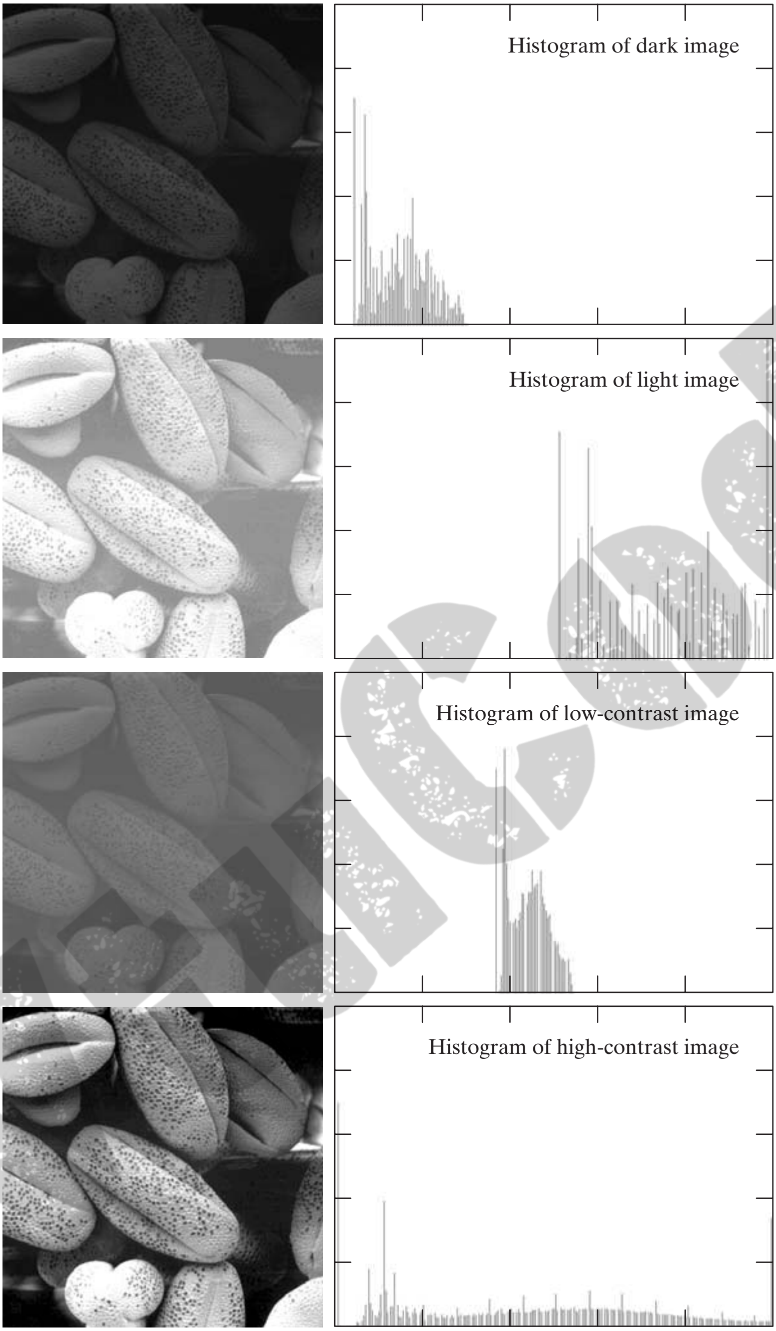


FIGURE 3.16 Four basic image types: dark, light, low contrast, high contrast, and their corresponding histograms.

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

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

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 $0 \leq r \leq L - 1$; and

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

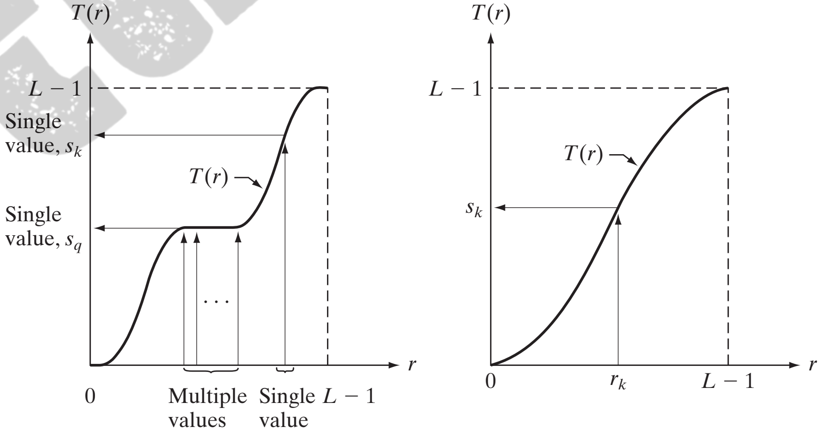
In some formulations to be discussed later, we use the inverse

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

in which case we change condition (a) to

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

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 $T(r_2) \geq T(r_1)$ 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.

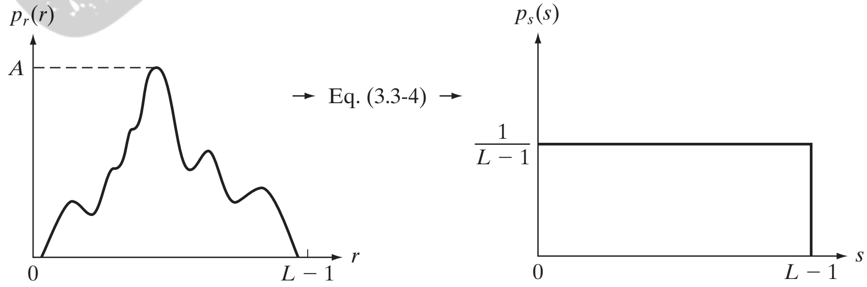
To find the $p_s(s)$ corresponding to the transformation just discussed, we use Eq. (3.3-3). We know from Leibniz's rule in basic calculus that the derivative of a definite integral with respect to its upper limit is the integrand evaluated at the limit. That is,

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

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

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

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

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

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,

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

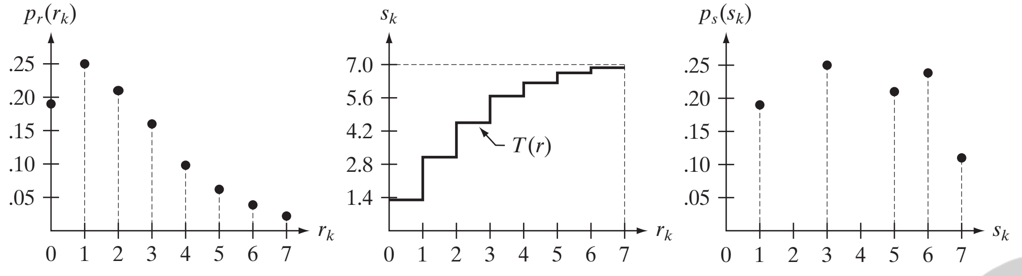
Similarly,

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

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:

$$
\begin{array}{ll}
s_0 = 1.33 \rightarrow 1 & s_4 = 6.23 \rightarrow 6 \\
s_1 = 3.08 \rightarrow 3 & s_5 = 6.65 \rightarrow 7 \\
s_2 = 4.55 \rightarrow 5 & s_6 = 6.86 \rightarrow 7 \\
s_3 = 5.67 \rightarrow 6 & s_7 = 7.00 \rightarrow 7
\end{array}
$$

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*.

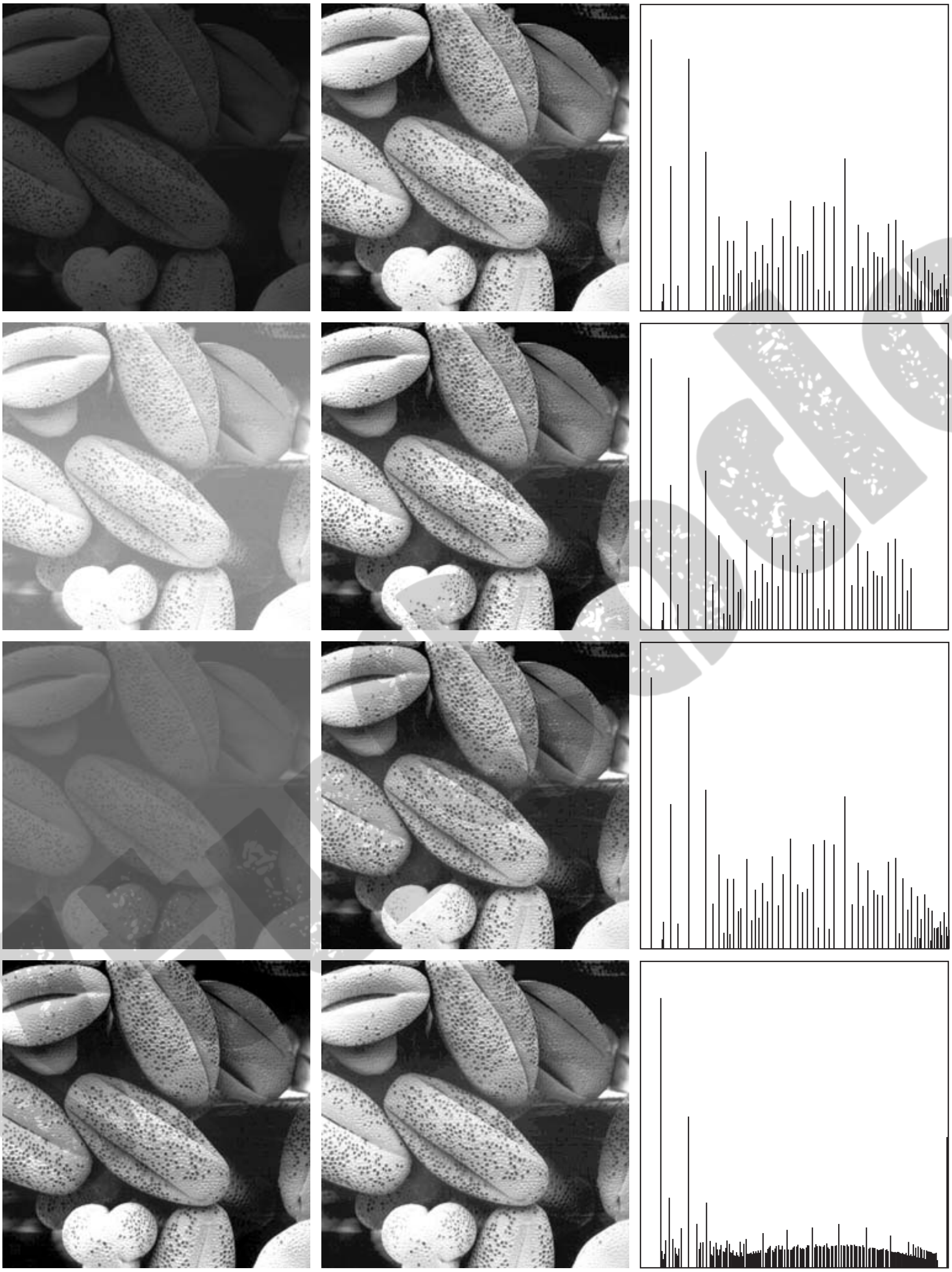


FIGURE 3.20 Left column: images from Fig. 3.16. Center column: corresponding histogram-equalized images. Right column: histograms of the images in the center column.

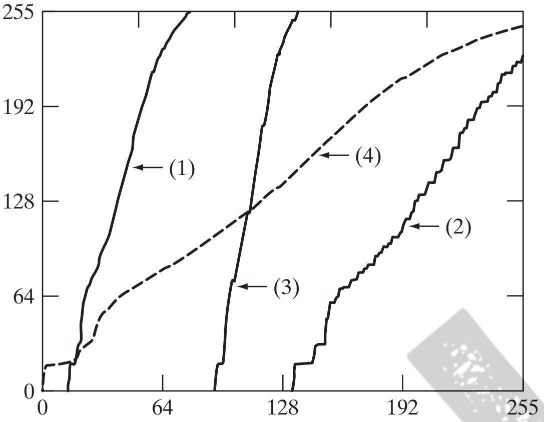


FIGURE 3.21 Transformation functions for histogram equalization. Transformations (1) through (4) were obtained from the histograms of the images (from top to bottom) in the left column of Fig. 3.20 using Eq. (3.3-8).

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.

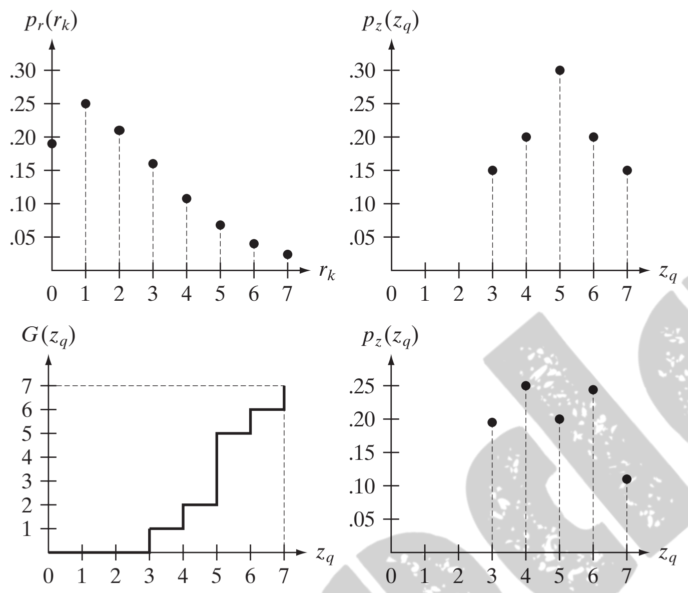
EXAMPLE 3.8: A simple example of histogram specification.

■ 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,

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

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:

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

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

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

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

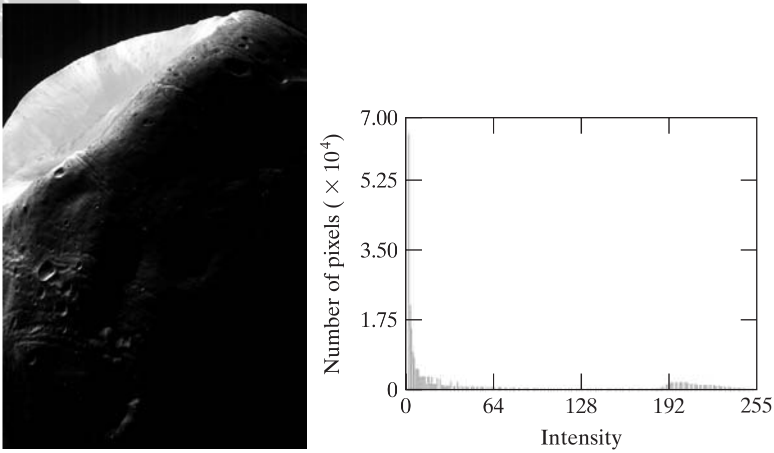
s_k	→	z_q
1	→	3
3	→	4
5	→	5
6	→	6
7	→	7

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.

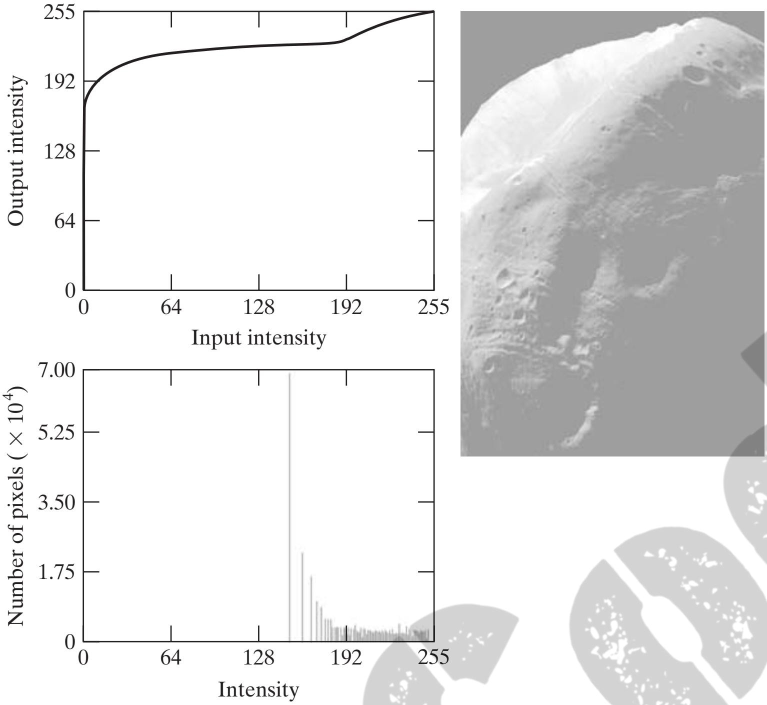
■ Figure 3.23(a) shows an image of the Mars moon, Phobos, taken by NASA's *Mars Global Surveyor*. Figure 3.23(b) shows the histogram of Fig. 3.23(a). The image is dominated by large, dark areas, resulting in a histogram characterized by a large concentration of pixels in the dark end of the gray scale. 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.)

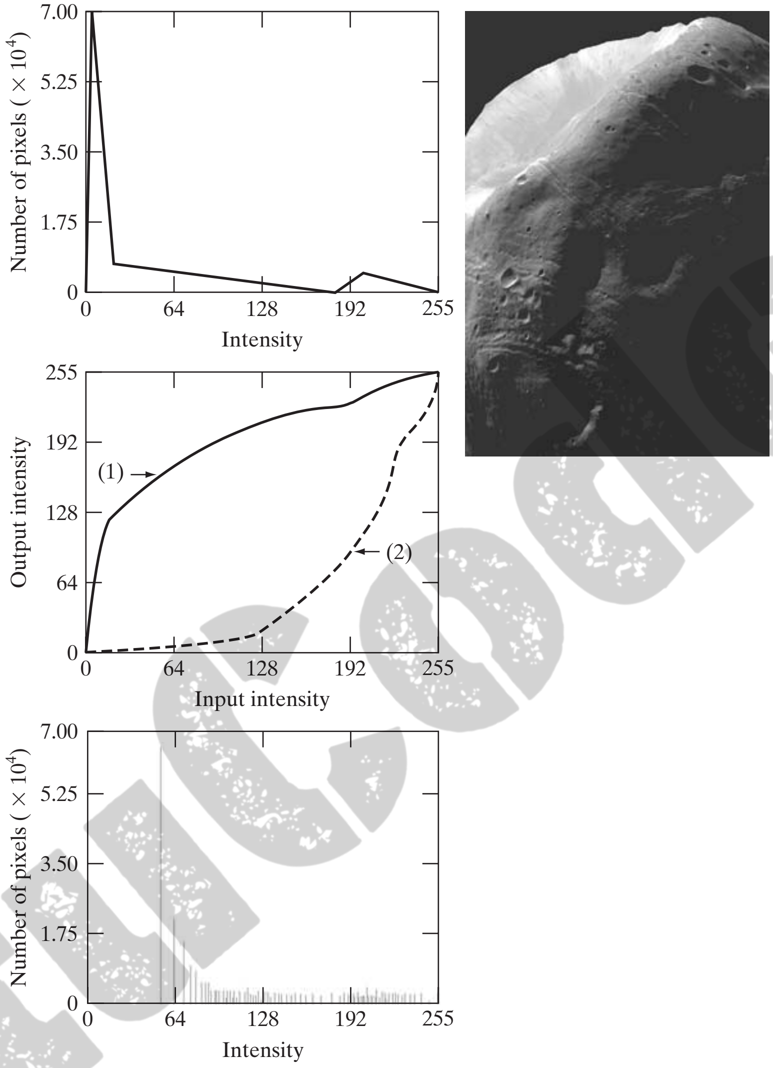


a b
c

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

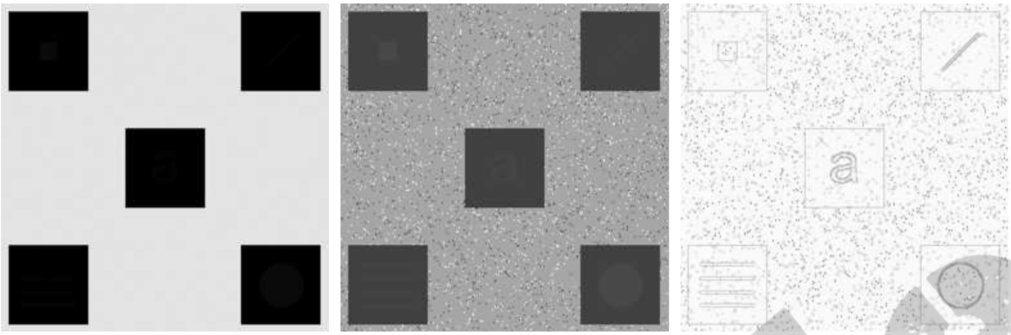


FIGURE 3.26 (a) Original image. (b) Result of global histogram equalization. (c) Result of local histogram equalization applied to (a), using a neighborhood of size 3 × 3.

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

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

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

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.

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

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

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

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

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.

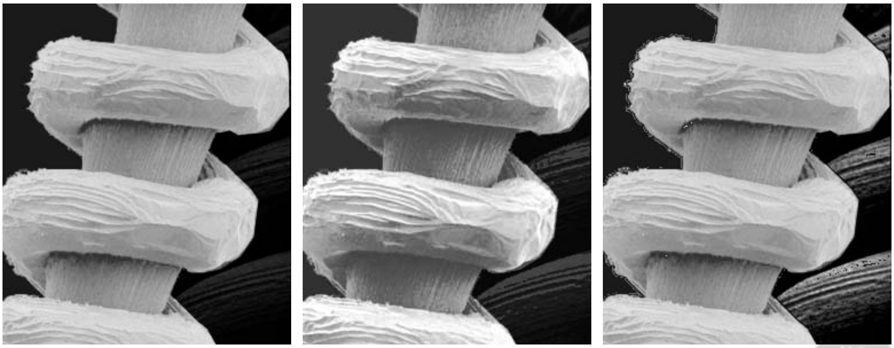


FIGURE 3.27 (a) SEM image of a tungsten filament magnified approximately 130×. (b) Result of global histogram equalization. (c) Image enhanced using local histogram statistics. (Original image courtesy of Mr. Michael Shaffer, Department of Geological Sciences, University of Oregon, Eugene.)

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:

$$\begin{aligned} g(x, y) = {} & w(-1,-1)f(x-1, y-1) + w(-1,0)f(x-1, y) + \dots \\ & + w(0,0)f(x, y) + \dots + w(1,1)f(x+1, y+1) \end{aligned}$$

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:

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.

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

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

[†] 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.

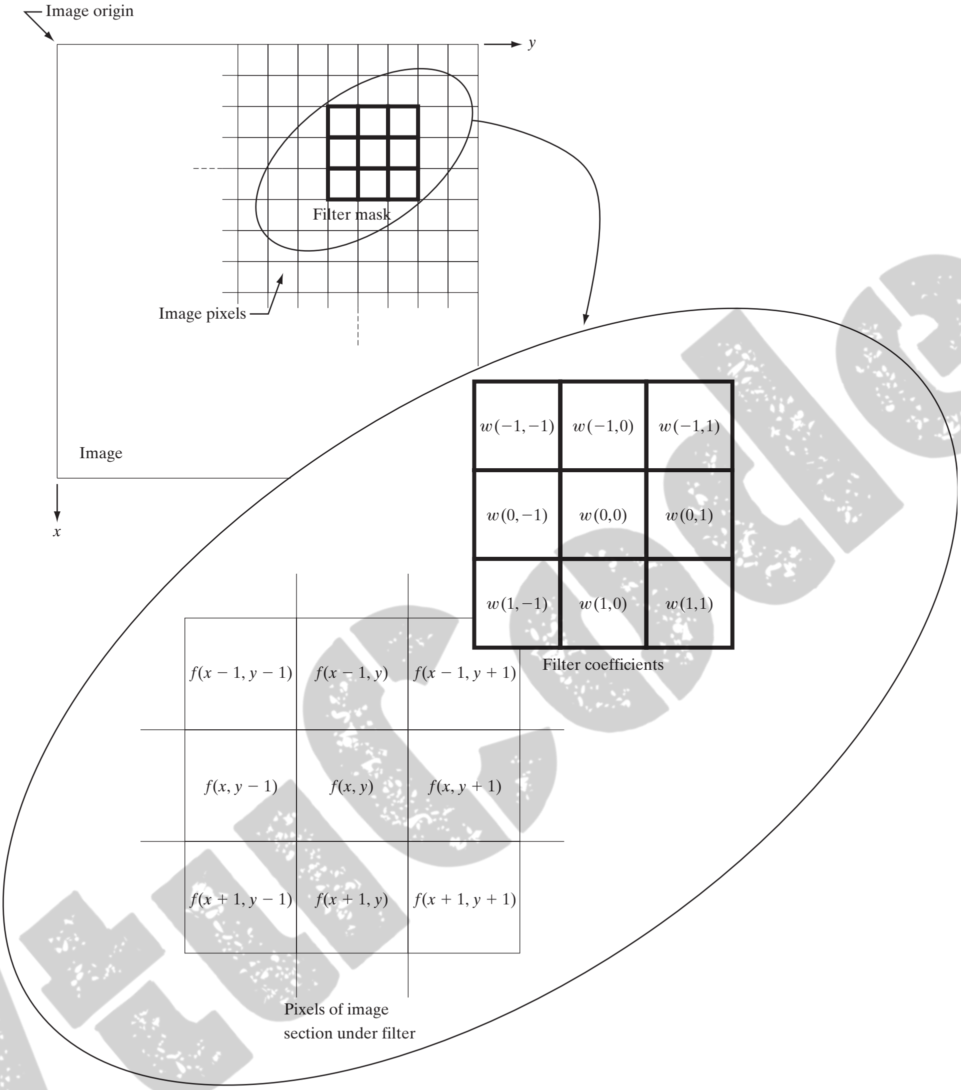


FIGURE 3.28 The mechanics of linear spatial filtering using a 3 × 3 filter mask. The form chosen to denote the coordinates of the filter mask coefficients simplifies writing expressions for linear filtering.

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

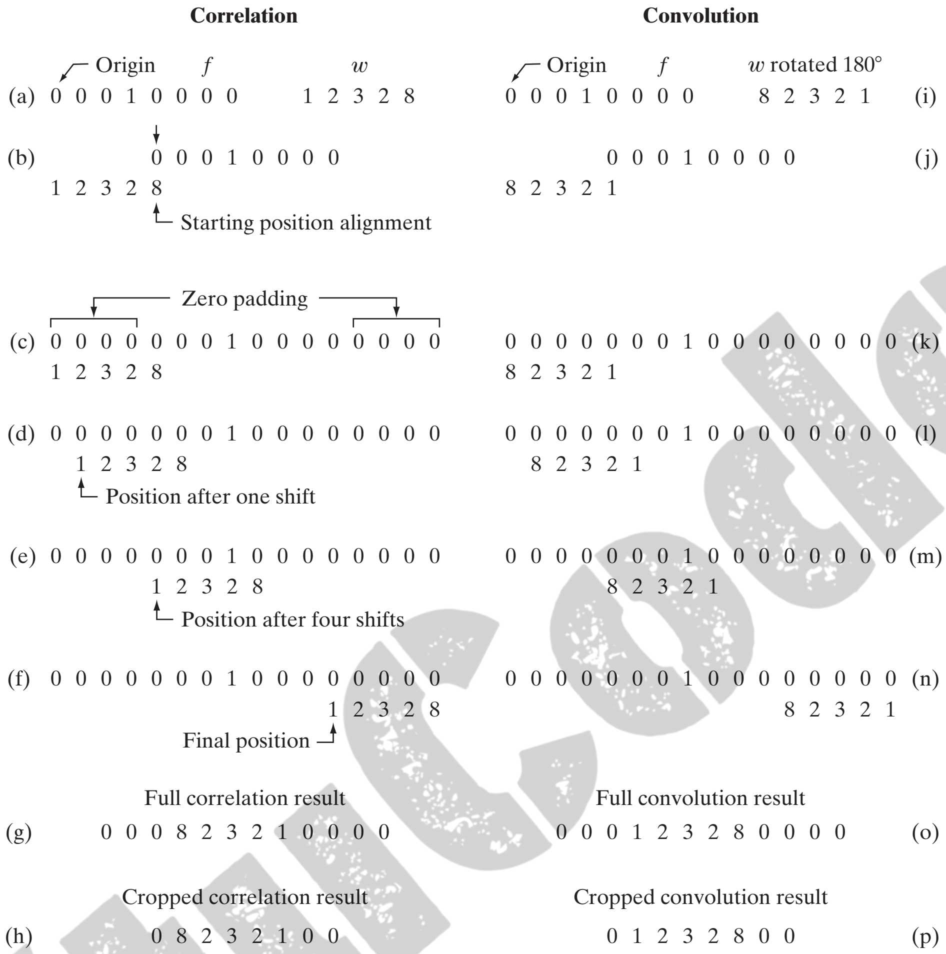

Correlation
Convolution
Origin
f
w
w rotated 180°
(a) 0 0 0 1 0 0 0 0 1 2 3 2 8
0 0 0 1 0 0 0 0 8 2 3 2 1 (i)
(b) 0 0 0 1 0 0 0 0
1 2 3 2 8
Starting position alignment
0 0 0 1 0 0 0 0 (j)
8 2 3 2 1
Zero padding
(c) 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
1 2 3 2 8
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 (k)
8 2 3 2 1
(d) 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
1 2 3 2 8
Position after one shift
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 (l)
8 2 3 2 1
(e) 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
1 2 3 2 8
Position after four shifts
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 (m)
8 2 3 2 1
(f) 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
1 2 3 2 8
Final position
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 (n)
8 2 3 2 1
Full correlation result
(g) 0 0 0 8 2 3 2 1 0 0 0 0
Full convolution result
0 0 0 1 2 3 2 8 0 0 0 0 (o)
Cropped correlation result
(h) 0 8 2 3 2 1 0 0
Cropped convolution result
0 1 2 3 2 8 0 0 (p)

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.

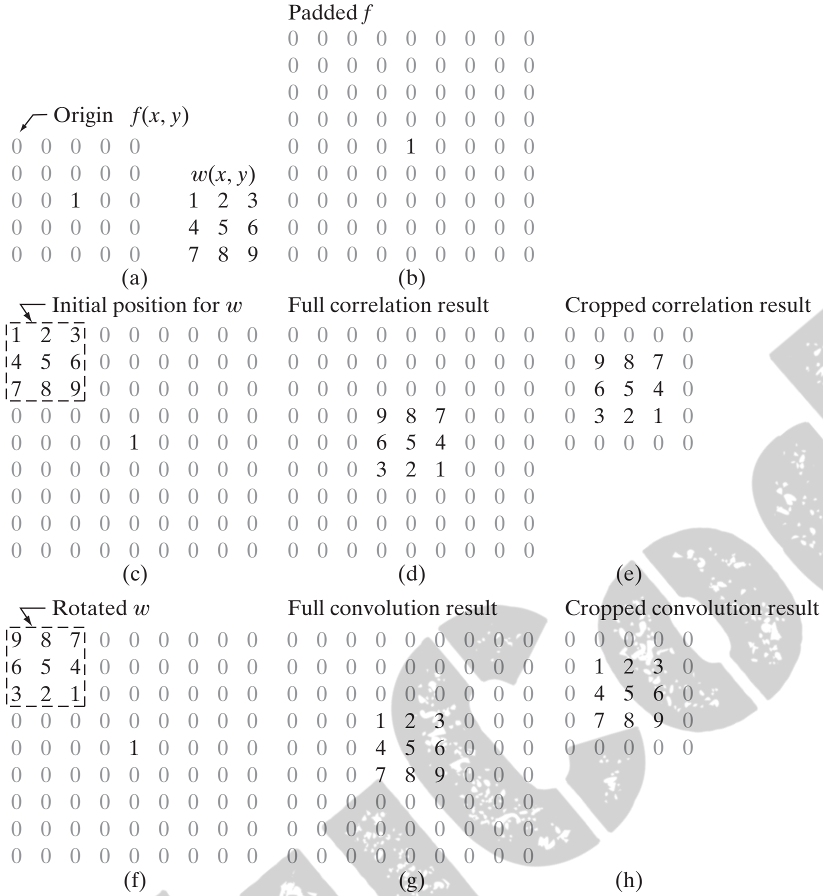
In 2-D, rotation by 180° is equivalent to flipping the mask along one axis and then the other.

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:

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

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.



(a) Origin $f(x, y)$, $w(x, y)$

$$f(x,y)=\begin{matrix}0&0&0&0&0\\0&0&0&0&0\\0&0&1&0&0\\0&0&0&0&0\\0&0&0&0&0\end{matrix}\qquad w(x,y)=\begin{matrix}1&2&3\\4&5&6\\7&8&9\end{matrix}$$

(b) Padded f

$$\begin{matrix}0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&1&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\end{matrix}$$

(c) Initial position for w

$$\begin{matrix}1&2&3&0&0&0&0&0&0\\4&5&6&0&0&0&0&0&0\\7&8&9&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&1&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\end{matrix}$$

(d) Full correlation result

$$\begin{matrix}0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&9&8&7&0&0&0\\0&0&0&6&5&4&0&0&0\\0&0&0&3&2&1&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\end{matrix}$$

(e) Cropped correlation result

$$\begin{matrix}0&0&0&0&0\\0&9&8&7&0\\0&6&5&4&0\\0&3&2&1&0\\0&0&0&0&0\end{matrix}$$

(f) Rotated w

$$\begin{matrix}9&8&7&0&0&0&0&0&0\\6&5&4&0&0&0&0&0&0\\3&2&1&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&1&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\end{matrix}$$

(g) Full convolution result

$$\begin{matrix}0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&1&2&3&0&0&0\\0&0&0&4&5&6&0&0&0\\0&0&0&7&8&9&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\end{matrix}$$

(h) Cropped convolution result

$$\begin{matrix}0&0&0&0&0\\0&1&2&3&0\\0&4&5&6&0\\0&7&8&9&0\\0&0&0&0&0\end{matrix}$$

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 $w(x, y) \bigstar f(x, y)$,[†] is given by the expression

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

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 $w(x, y) \bigstar f(x, y)$. 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 $w(x, y) \bigstar f(x, y) = f(x, y) \bigstar w(x, y)$.

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.

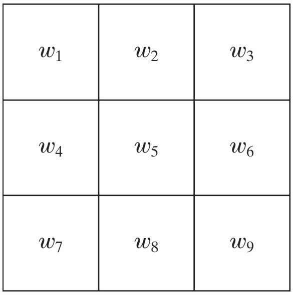
3.4.3 Vector Representation of Linear Filtering

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

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

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

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

where **w** and **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 $z_i, i = 1, 2, \dots, 9$, 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,

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

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

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

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.

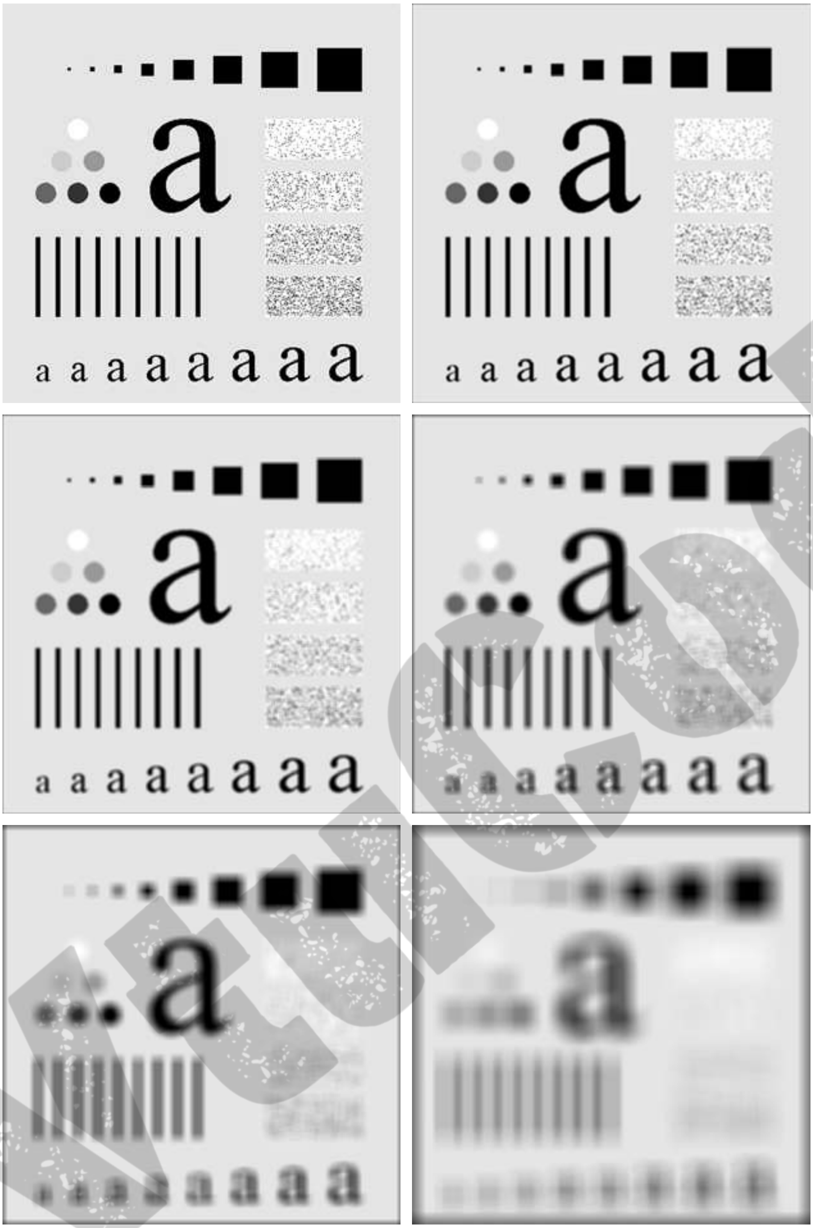


FIGURE 3.33 (a) Original image, of size 500 × 500 pixels. (b)–(f) Results of smoothing with square averaging filter masks of sizes $m = 3, 5, 9, 15$, and 35, respectively. The black squares at the top are of sizes 3, 5, 9, 15, 25, 35, 45, and 55 pixels, respectively; their borders are 25 pixels apart. 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.

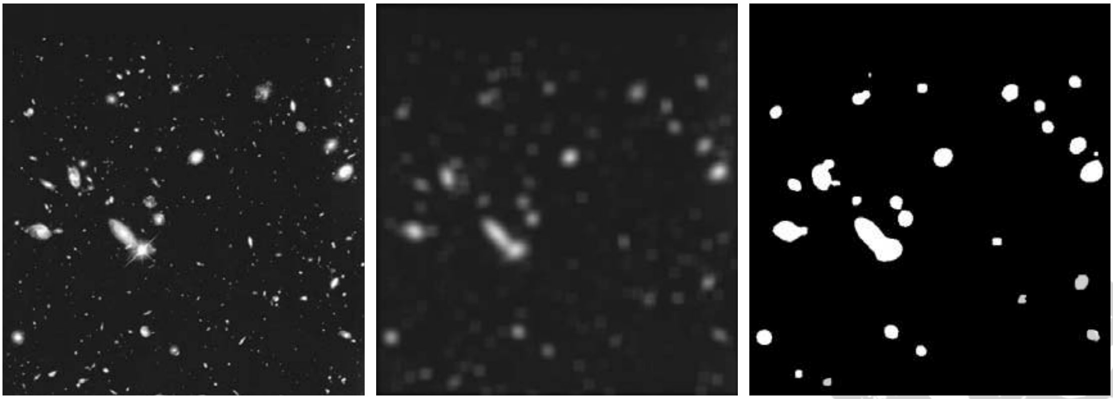


FIGURE 3.34 (a) Image of size 528 × 485 pixels from the Hubble Space Telescope. (b) Image filtered with a 15 × 15 averaging mask. (c) Result of thresholding (b). (Original image courtesy of NASA.)

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.

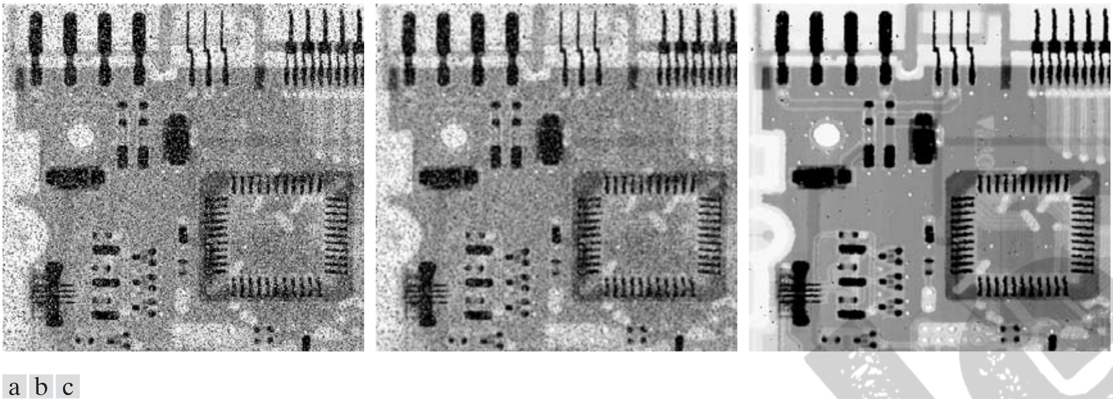


FIGURE 3.35 (a) X-ray image of circuit board corrupted by salt-and-pepper noise. (b) Noise reduction with a 3 × 3 averaging mask. (c) Noise reduction with a 3 × 3 median filter. (Original image courtesy of Mr. Joseph E. Pascente, Lixi, Inc.)

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 $R = \max\{z_k | k = 1, 2, \dots, 9\}$. 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

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.

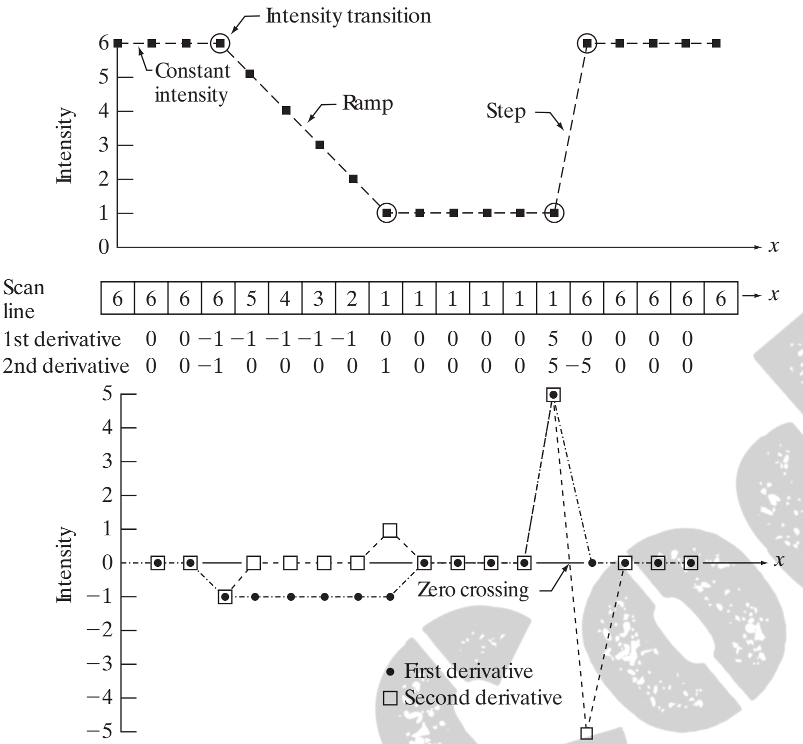
$$\frac{\partial f}{\partial x} = f(x + 1) - f(x) \tag{3.6-1}$$

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



Scan line	6	6	6	6	5	4	3	2	1	1	1	1	1	1	6	6	6	6	6 → x
1st derivative		0	0	−1	−1	−1	−1	−1	0	0	0	0	0	5	0	0	0	0	
2nd derivative		0	0	−1	0	0	0	0	1	0	0	0	0	5	−5	0	0	0	

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

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

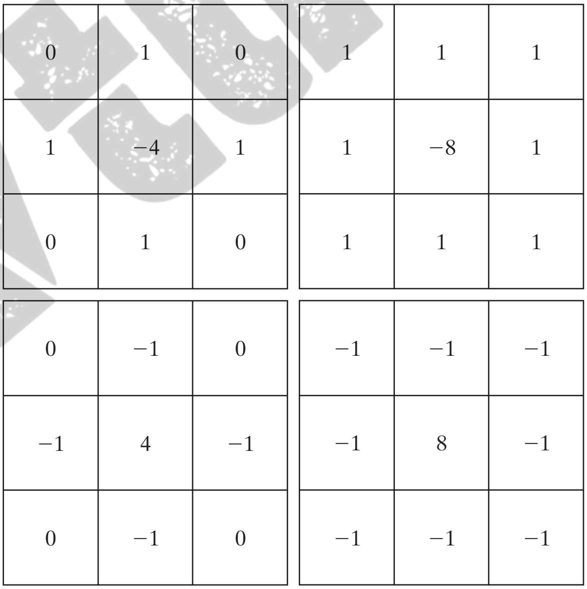
Therefore, it follows from the preceding three equations that the discrete Laplacian of two variables is

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

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	−4	1
0	1	0

1	1	1
1	−8	1
1	1	1

0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−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.

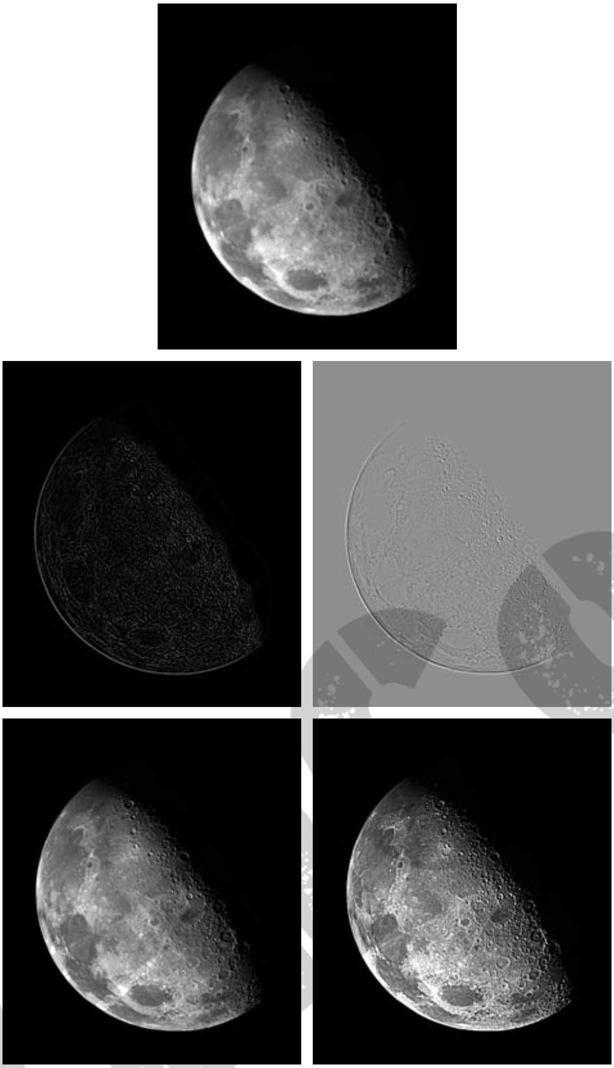
■ Figure 3.38(a) shows a slightly blurred image of the North Pole of the moon. Figure 3.38(b) shows the result of filtering this image with the Laplacian mask in Fig. 3.37(a). Large sections of this image are black because the Laplacian contains both positive and negative values, and all negative values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

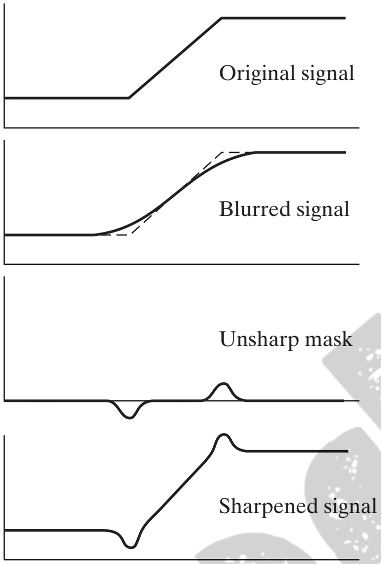
Letting $\bar{f}(x, y)$ denote the blurred image, unsharp masking is expressed in equation form as follows. First we obtain the mask:

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

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

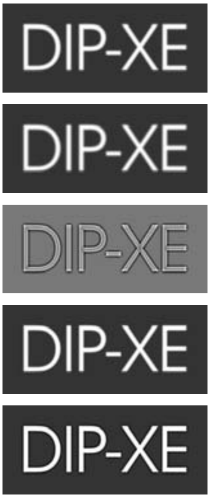


FIGURE 3.40 (a) Original image. (b) Result of blurring with a Gaussian filter. (c) Unsharp mask. (d) Result of using unsharp masking. (e) Result of using highboost filtering.

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

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

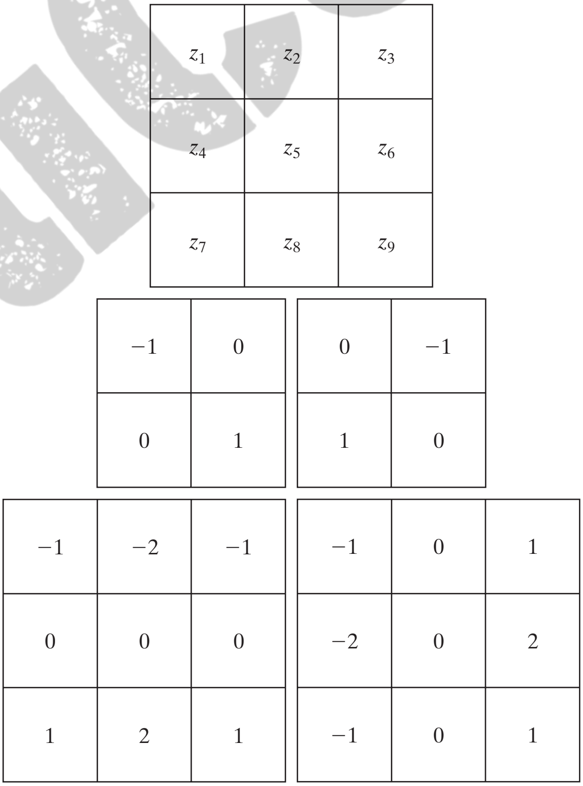
is the *value* at (x, y) of the rate of change in the direction of the gradient vector. Note that $M(x, y)$ is an image of the same size as the original, created when x and y are allowed to vary over all pixel locations in f. It is common practice to refer to this image as the *gradient image* (or simply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

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

b

−1	0
0	1

c

0	−1
1	0

d

−1	−2	−1
0	0	0
1	2	1

e

−1	0	1
−2	0	2
−1	0	1

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

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

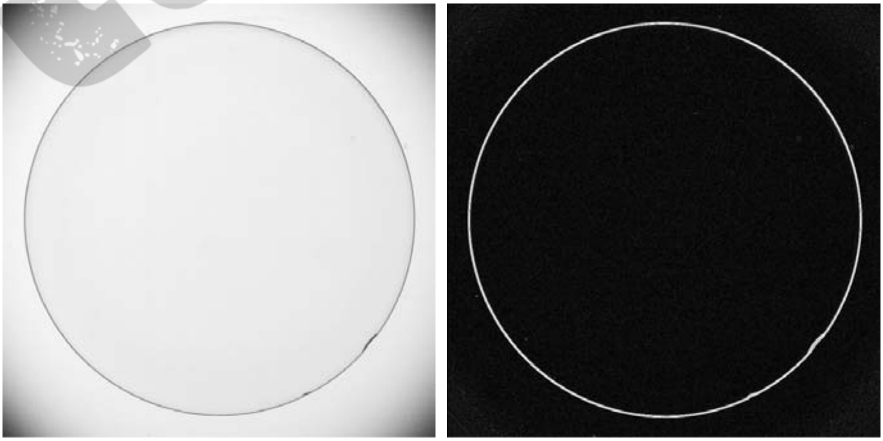
The masks in Figs. 3.41(d) and (e) are called the *Sobel operators*. The idea behind using a weight value of 2 in the center coefficient is to achieve some smoothing by giving more importance to the center point (we discuss this in more detail in Chapter 10). Note that the coefficients in all the masks 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.)

4.2 Preliminary Concepts

In order to simplify the progression of ideas presented in this chapter, we pause briefly to introduce several of the basic concepts that underlie the material that follows in later sections.

4.2.1 Complex Numbers

A complex number, C, is defined as

$$C = R + jI \tag{4.2-1}$$

where R and I are real numbers, and j is an imaginary number equal to the square of -1; that is, $j = \sqrt{-1}$. Here, R denotes the *real part* of the complex number and I its *imaginary part*. Real numbers are a subset of complex numbers in which $I = 0$. The *conjugate* of a complex number C, denoted C^*, is defined as

$$C^* = R - jI \tag{4.2-2}$$

Complex numbers can be viewed geometrically as points in a plane (called the *complex plane*) whose abscissa is the *real axis* (values of R) and whose ordinate is the *imaginary axis* (values of I). That is, the complex number $R + jI$ is point (R, I) in the rectangular coordinate system of the complex plane.

Sometimes, it is useful to represent complex numbers in polar coordinates,

$$C = |C|(\cos\theta + j\sin\theta) \tag{4.2-3}$$

where $|C| = \sqrt{R^2 + I^2}$ is the length of the vector extending from the origin of the complex plane to point (R, I), and θ is the angle between the vector and the real axis. Drawing a simple diagram of the real and complex axes with the vector in the first quadrant will reveal that $\tan\theta = (I/R)$ or $\theta = \arctan(I/R)$. The arctan function returns angles in the range $[-\pi/2, \pi/2]$. However, because I and R can be positive and negative independently, we need to be able to obtain angles in the full range $[-\pi, \pi]$. This is accomplished simply by keeping track of the sign of I and R when computing θ. Many programming languages do this automatically via so called *four-quadrant arctangent* functions. For example, MATLAB provides the function `atan2(Imag, Real)` for this purpose.

Using Euler's formula,

$$e^{j\theta} = \cos\theta + j\sin\theta \tag{4.2-4}$$

where $e = 2.71828\ldots$, gives the following familiar representation of complex numbers in polar coordinates,

$$C = |C|e^{j\theta} \tag{4.2-5}$$

where $|C|$ and θ are as defined above. For example, the polar representation of the complex number $1 + j2$ is $\sqrt{3}e^{j\theta}$, where $\theta = 64.4°$ or 1.1 radians. The preceding equations are applicable also to complex functions. For example, a complex function, $F(u)$, of a variable u, can be expressed as the sum $F(u) = R(u) + jI(u)$, where $R(u)$ and $I(u)$ are the real and imaginary component functions. As previously noted, the complex conjugate is $F^*(u) = R(u) - jI(u)$, the magnitude is $|F(u)| = \sqrt{R(u)^2 + I(u)^2}$, and the angle is $\theta(u) = \arctan[I(u)/R(u)]$. We return to complex functions several times in the course of this and the next chapter.

4.2.2 Fourier Series

As indicated in Section 4.1.1, a function $f(t)$ of a continuous variable t that is periodic with period, T, can be expressed as the sum of sines and cosines multiplied by appropriate coefficients. This sum, known as a *Fourier series*, has the form

$$f(t) = \sum_{n=-\infty}^{\infty} c_n e^{j\frac{2\pi n}{T}t} \tag{4.2-6}$$

where

$$c_n = \frac{1}{T}\int_{-T/2}^{T/2} f(t)e^{-j\frac{2\pi n}{T}t}\,dt \qquad \text{for } n = 0, \pm1, \pm2, \ldots \tag{4.2-7}$$

are the coefficients. The fact that Eq. (4.2-6) is an expansion of sines and cosines follows from Euler's formula, Eq. (4.2-4). We will return to the Fourier series later in this section.

4.2.3 Impulses and Their Sifting Property

Central to the study of linear systems and the Fourier transform is the concept of an impulse and its sifting property. A *unit impulse* of a continuous variable t located at $t = 0$, denoted $\delta(t)$, is *defined* as

$$\delta(t) = \begin{cases} \infty & \text{if } t = 0 \\ 0 & \text{if } t \neq 0 \end{cases} \tag{4.2-8a}$$

and is constrained also to satisfy the identity

$$\int_{-\infty}^{\infty} \delta(t)\,dt = 1 \tag{4.2-8b}$$

An impulse is not a function in the usual sense. A more accurate name is a *distribution* or *generalized function*. However, one often finds in the literature the names *impulse function*, *delta function*, and *Dirac delta function*, despite the misnomer.

Physically, if we interpret t as time, an impulse may be viewed as a spike of infinity amplitude and zero duration, having unit area. An impulse has the so-called *sifting property* with respect to integration,

$$\int_{-\infty}^{\infty} f(t)\,\delta(t)\,dt = f(0) \tag{4.2-9}$$

To *sift* means literally to separate, or to separate out by putting through a sieve.

provided that $f(t)$ is continuous at $t = 0$, a condition typically satisfied in practice. Sifting simply yields the *value* of the function $f(t)$ at the *location* of the impulse (i.e., the origin, $t = 0$, in the previous equation). A more general statement of the sifting property involves an impulse located at an arbitrary point t_0, denoted by $\delta(t - t_0)$. In this case, the sifting property becomes

$$\int_{-\infty}^{\infty} f(t)\,\delta(t - t_0)\,dt = f(t_0) \tag{4.2-10}$$

which yields the value of the function at the impulse location, t_0. For instance, if $f(t) = \cos(t)$, using the impulse $\delta(t - \pi)$ in Eq. (4.2-10) yields the result $f(\pi) = \cos(\pi) = -1$. The power of the sifting concept will become quite evident shortly.

Let x represent a *discrete* variable. The *unit discrete impulse*, $\delta(x)$, serves the same purposes in the context of discrete systems as the impulse $\delta(t)$ does when working with continuous variables. It is defined as

$$\delta(x) = \begin{cases} 1 & x = 0 \\ 0 & x \neq 0 \end{cases} \tag{4.2-11a}$$

Clearly, this definition also satisfies the discrete equivalent of Eq. (4.2-8b):

$$\sum_{x=-\infty}^{\infty} \delta(x) = 1 \tag{4.2-11b}$$

The sifting property for discrete variables has the form

$$\sum_{x=-\infty}^{\infty} f(x)\,\delta(x) = f(0) \tag{4.2-12}$$

or, more generally using a discrete impulse located at $x = x_0$,

$$\sum_{x=-\infty}^{\infty} f(x)\,\delta(x - x_0) = f(x_0) \tag{4.2-13}$$

As before, we see that the sifting property simply yields the value of the function at the location of the impulse. Figure 4.2 shows the unit discrete impulse diagrammatically. Unlike its continuous counterpart, the discrete impulse is an ordinary function.

Of particular interest later in this section is an *impulse train*, $s_{\Delta T}(t)$, defined as the sum of infinitely many *periodic* impulses ΔT units apart:

$$s_{\Delta T}(t) = \sum_{n=-\infty}^{\infty} \delta(t - n\Delta T) \tag{4.2-14}$$

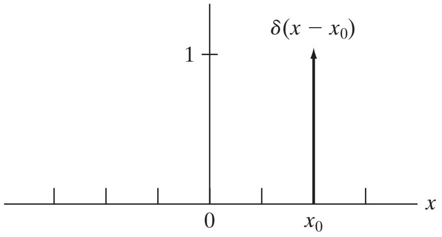


FIGURE 4.2 A unit discrete impulse located at $x = x_0$. Variable x is discrete, and δ is 0 everywhere except at $x = x_0$.

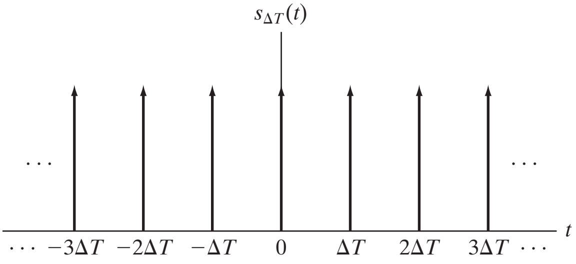


FIGURE 4.3 An impulse train.

Figure 4.3 shows an impulse train. The impulses can be continuous or discrete.

4.2.4 The Fourier Transform of Functions of One Continuous Variable

The *Fourier transform* of a continuous function $f(t)$ of a continuous variable, t, denoted $\Im\{f(t)\}$, is *defined* by the equation[†]

$$\Im\{f(t)\} = \int_{-\infty}^{\infty} f(t)\,e^{-j2\pi\mu t}\,dt \tag{4.2-15}$$

where μ is also a continuous variable. Because t is integrated out, $\Im\{f(t)\}$ is a function only of μ. We denote this fact explicitly by writing the Fourier transform as $\Im\{f(t)\} = F(\mu)$; that is, the Fourier transform of $f(t)$ may be written for convenience as

$$F(\mu) = \int_{-\infty}^{\infty} f(t)\,e^{-j2\pi\mu t}\,dt \tag{4.2-16}$$

Conversely, given $F(\mu)$, we can obtain $f(t)$ back using the *inverse Fourier transform*, $f(t) = \Im^{-1}\{F(\mu)\}$, written as

$$f(t) = \int_{-\infty}^{\infty} F(\mu)\,e^{j2\pi\mu t}\,d\mu \tag{4.2-17}$$

where we made use of the fact that variable μ is integrated out in the inverse transform and wrote simple $f(t)$, rather than the more cumbersome notation $f(t) = \Im^{-1}\{F(\mu)\}$. Equations (4.2-16) and (4.2-17) comprise the so-called *Fourier transform pair*. They indicate the important fact mentioned in Section 4.1 that a function can be recovered from its transform.

Using Euler's formula we can express Eq. (4.2-16) as

$$F(\mu) = \int_{-\infty}^{\infty} f(t)\big[\cos(2\pi\mu t) - j\sin(2\pi\mu t)\big]\,dt \tag{4.2-18}$$

[†] Conditions for the existence of the Fourier transform are complicated to state in general (Champeney [1987]), but a sufficient condition for its existence is that the integral of the absolute value of $f(t)$, or the integral of the square of $f(t)$, be finite. Existence is seldom an issue in practice, except for idealized signals, such as sinusoids that extend forever. These are handled using generalized impulse functions. Our primary interest is in the discrete Fourier transform pair which, as you will see shortly, is guaranteed to exist for all finite functions.

If $f(t)$ is real, we see that its transform in general is complex. Note that the Fourier transform is an expansion of $f(t)$ multiplied by sinusoidal terms whose frequencies are determined by the values of μ (variable t is integrated out, as mentioned earlier). Because the only variable left after integration is frequency, we say that the domain of the Fourier transform is the *frequency domain*. We discuss the frequency domain and its properties in more detail later in this chapter. In our discussion, t can represent any continuous variable, and the units of the frequency variable μ depend on the units of t. For example, if t represents time in seconds, the units of μ are cycles/sec or Hertz (Hz). If t represents distance in meters, then the units of μ are cycles/meter, and so on. In other words, the units of the frequency domain are cycles per unit of the independent variable of the input function.

For consistency in terminology used in the previous two chapters, and to be used later in this chapter in connection with images, we refer to the domain of variable t in general as the *spatial domain*.

EXAMPLE 4.1: Obtaining the Fourier transform of a simple function.

■ The Fourier transform of the function in Fig. 4.4(a) follows from Eq. (4.2-16):

$$\begin{aligned}F(\mu) &= \int_{-\infty}^{\infty} f(t)e^{-j2\pi\mu t}\,dt = \int_{-W/2}^{W/2} Ae^{-j2\pi\mu t}\,dt\\ &= \frac{-A}{j2\pi\mu}\Big[e^{-j2\pi\mu t}\Big]_{-W/2}^{W/2} = \frac{-A}{j2\pi\mu}\Big[e^{-j\pi\mu W} - e^{j\pi\mu W}\Big]\\ &= \frac{A}{j2\pi\mu}\Big[e^{j\pi\mu W} - e^{-j\pi\mu W}\Big]\\ &= AW\frac{\sin(\pi\mu W)}{(\pi\mu W)}\end{aligned}$$

where we used the trigonometric identity $\sin\theta = (e^{j\theta} - e^{-j\theta})/2j$. In this case the complex terms of the Fourier transform combined nicely into a real sine

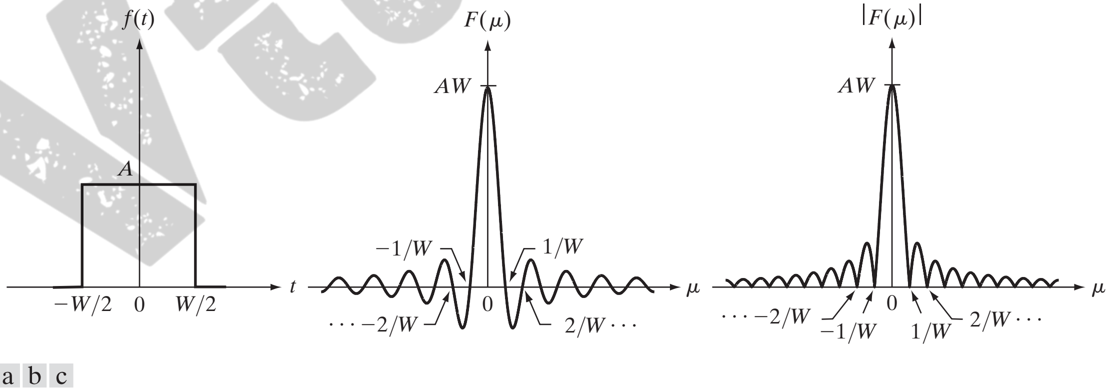


FIGURE 4.4 (a) A simple function; (b) its Fourier transform; and (c) the spectrum. All functions extend to infinity in both directions.

function. The result in the last step of the preceding expression is known as the *sinc* function:

$$\operatorname{sinc}(m) = \frac{\sin(\pi m)}{(\pi m)} \tag{4.2-19}$$

where $\operatorname{sinc}(0) = 1$, and $\operatorname{sinc}(m) = 0$ for all other *integer* values of m. Figure 4.4(b) shows a plot of $F(\mu)$.

In general, the Fourier transform contains complex terms, and it is customary for display purposes to work with the magnitude of the transform (a real quantity), which is called the *Fourier spectrum* or the *frequency spectrum*:

$$|F(\mu)| = AT\left|\frac{\sin(\pi\mu W)}{(\pi\mu W)}\right|$$

Figure 4.4(c) shows a plot of $|F(\mu)|$ as a function of frequency. The key properties to note are that the locations of the zeros of both $F(\mu)$ and $|F(\mu)|$ are *inversely* proportional to the width, W, of the "box" function, that the height of the lobes decreases as a function of distance from the origin, and that the function extends to infinity for both positive and negative values of μ. As you will see later, these properties are quite helpful in interpreting the spectra of two-dimensional Fourier transforms of images. ■

EXAMPLE 4.2: Fourier transform of an impulse and of an impulse train.

■ The Fourier transform of a unit impulse located at the origin follows from Eq. (4.2-16):

$$\begin{aligned} F(\mu) &= \int_{-\infty}^{\infty} \delta(t)\, e^{-j2\pi\mu t}\, dt \\ &= \int_{-\infty}^{\infty} e^{-j2\pi\mu t}\, \delta(t)\, dt \\ &= e^{-j2\pi\mu 0} = e^{0} \\ &= 1 \end{aligned}$$

where the third step follows from the sifting property in Eq. (4.2-9). Thus, we see that the Fourier transform of an impulse located at the origin of the spatial domain is a constant in the frequency domain. Similarly, the Fourier transform of an impulse located at $t = t_0$ is

$$\begin{aligned} F(\mu) &= \int_{-\infty}^{\infty} \delta(t - t_0)\, e^{-j2\pi\mu t}\, dt \\ &= \int_{-\infty}^{\infty} e^{-j2\pi\mu t}\, \delta(t - t_0)\, dt \\ &= e^{-j2\pi\mu t_0} \\ &= \cos(2\pi\mu t_0) - j\sin(2\pi\mu t_0) \end{aligned}$$

where the third line follows from the sifting property in Eq. (4.2-10) and the last line follows from Euler's formula. These last two lines are equivalent representations of a unit circle centered on the origin of the complex plane.

In Section 4.3, we make use of the Fourier transform of a periodic impulse train. Obtaining this transform is not as straightforward as we just showed for individual impulses. However, understanding how to derive the transform of an impulse train is quite important, so we take the time to derive it in detail here. We start by noting that the only difference in the *form* of Eqs. (4.2-16) and (4.2-17) is the sign of the exponential. Thus, if a function $f(t)$ has the Fourier transform $F(\mu)$, then the latter function evaluated at t, that is, $F(t)$, must have the transform $f(-\mu)$. Using this *symmetry* property and given, as we showed above, that the Fourier transform of an impulse $\delta(t - t_0)$ is $e^{-j2\pi\mu t_0}$, it follows that the function $e^{-j2\pi t_0 t}$ has the transform $\delta(-\mu - t_0)$. By letting $-t_0 = a$, it follows that the transform of $e^{j2\pi at}$ is $\delta(-\mu + a) = \delta(\mu - a)$, where the last step is true because δ is not zero only when $\mu = a$, which is the same result for either $\delta(-\mu + a)$ or $\delta(\mu - a)$, so the two forms are equivalent.

The impulse train $s_{\Delta T}(t)$ in Eq. (4.2-14) is periodic with period ΔT, so we know from Section 4.2.2 that it can be expressed as a Fourier series:

$$s_{\Delta T}(t) = \sum_{n=-\infty}^{\infty} c_n e^{j\frac{2\pi n}{\Delta T}t}$$

where

$$c_n = \frac{1}{\Delta T}\int_{-\Delta T/2}^{\Delta T/2} s_{\Delta T}(t)\, e^{-j\frac{2\pi n}{\Delta T}t}\, dt$$

With reference to Fig. 4.3, we see that the integral in the interval $[-\Delta T/2, \Delta T/2]$ encompasses only the impulse of $s_{\Delta T}(t)$ that is located at the origin. Therefore, the preceding equation becomes

$$\begin{aligned} c_n &= \frac{1}{\Delta T}\int_{-\Delta T/2}^{\Delta T/2} \delta(t)\, e^{-j\frac{2\pi n}{\Delta T}t}\, dt \\ &= \frac{1}{\Delta T} e^0 \\ &= \frac{1}{\Delta T} \end{aligned}$$

The Fourier series expansion then becomes

$$s_{\Delta T}(t) = \frac{1}{\Delta T}\sum_{n=-\infty}^{\infty} e^{j\frac{2\pi n}{\Delta T}t}$$

Our objective is to obtain the Fourier transform of this expression. Because summation is a linear process, obtaining the Fourier transform of a sum is the same as obtaining the sum of the transforms of the individual components. These components are exponentials, and we established earlier in this example that

$$\Im\left\{e^{j\frac{2\pi n}{\Delta T}t}\right\} = \delta\left(\mu - \frac{n}{\Delta T}\right)$$

So, $S(\mu)$, the Fourier transform of the periodic impulse train $s_{\Delta T}(t)$, is

$$\begin{aligned} S(\mu) &= \Im\left\{s_{\Delta T}(t)\right\} \\ &= \Im\left\{\frac{1}{\Delta T}\sum_{n=-\infty}^{\infty} e^{j\frac{2\pi n}{\Delta T}t}\right\} \\ &= \frac{1}{\Delta T}\Im\left\{\sum_{n=-\infty}^{\infty} e^{j\frac{2\pi n}{\Delta T}t}\right\} \\ &= \frac{1}{\Delta T}\sum_{n=-\infty}^{\infty}\delta\left(\mu - \frac{n}{\Delta T}\right) \end{aligned}$$

This fundamental result tells us that the Fourier transform of an impulse train with period ΔT *is also an impulse train*, whose period is $1/\Delta T$. This inverse proportionality between the periods of $s_{\Delta T}(t)$ and $S(\mu)$ is analogous to what we found in Fig. 4.4 in connection with a box function and its transform. This property plays a fundamental role in the remainder of this chapter. ■

4.2.5 Convolution

We need one more building block before proceeding. We introduced the idea of convolution in Section 3.4.2. You learned in that section that convolution of two functions involves flipping (rotating by 180°) one function about its origin and sliding it past the other. At each displacement in the sliding process, we perform a computation, which in the case of Chapter 3 was a sum of products. In the present discussion, we are interested in the convolution of two continuous functions, $f(t)$ and $h(t)$, of one *continuous* variable, t, so we have to use integration instead of a summation. The convolution of these two functions, denoted as before by the operator $\bigstar$, is *defined* as

$$f(t) \bigstar h(t) = \int_{-\infty}^{\infty} f(\tau)\,h(t-\tau)\,d\tau \qquad (4.2\text{-}20)$$

where the minus sign accounts for the flipping just mentioned, t is the *displacement* needed to slide one function past the other, and τ is a dummy variable that is integrated out. We assume for now that the functions extend from $-\infty$ to ∞.

We illustrated the basic mechanics of convolution in Section 3.4.2, and we will do so again later in this chapter and in Chapter 5. At the moment, we are interested in finding the Fourier transform of Eq. (4.2-20). We start with Eq. (4.2-15):

$$\begin{aligned}\Im\{f(t)\star h(t)\} &= \int_{-\infty}^{\infty}\left[\int_{-\infty}^{\infty} f(\tau)\,h(t-\tau)\,d\tau\right]e^{-j2\pi\mu t}\,dt \\ &= \int_{-\infty}^{\infty} f(\tau)\left[\int_{-\infty}^{\infty} h(t-\tau)\,e^{-j2\pi\mu t}\,dt\right]d\tau\end{aligned}$$

The term inside the brackets is the Fourier transform of $h(t - \tau)$. We show later in this chapter that $\Im\{h(t - \tau)\} = H(\mu)\mathrm{e}^{-j2\pi\mu\tau}$, where $H(\mu)$ is the Fourier transform of $h(t)$. Using this fact in the preceding equation gives us

The same result would be obtained if the order of $f(t)$ and $h(t)$ were reversed, so convolution is commutative.

$$\begin{aligned}\Im\{f(t)\star h(t)\} &= \int_{-\infty}^{\infty} f(\tau)\left[H(\mu)\,e^{-j2\pi\mu\tau}\right]d\tau \\ &= H(\mu)\int_{-\infty}^{\infty} f(\tau)\,e^{-j2\pi\mu\tau}\,d\tau \\ &= H(\mu)\,F(\mu)\end{aligned}$$

Recalling from Section 4.2.4 that we refer to the domain of t as the spatial domain, and the domain of μ as the frequency domain, the preceding equation tells us that the Fourier transform of the convolution of two functions in the spatial domain is equal to the product in the frequency domain of the Fourier transforms of the two functions. Conversely, if we have the product of the two transforms, we can obtain the convolution in the spatial domain by computing the inverse Fourier transform. In other words, $f(t)\star h(t)$ and $H(u)\,F(u)$ are a Fourier transform pair. This result is one-half of the *convolution theorem* and is written as

$$f(t)\star h(t) \Leftrightarrow H(\mu)\,F(\mu) \tag{4.2-21}$$

The double arrow is used to indicate that the expression on the right is obtained by taking the Fourier transform of the expression on the left, while the expression on the left is obtained by taking the *inverse* Fourier transform of the expression on the right.

Following a similar development would result in the other half of the convolution theorem:

$$f(t)h(t) \Leftrightarrow H(\mu)\star F(\mu) \tag{4.2-22}$$

which states that convolution in the frequency domain is analogous to multiplication in the spatial domain, the two being related by the forward and inverse Fourier transforms, respectively. As you will see later in this chapter, the convolution theorem is the foundation for filtering in the frequency domain.

4.5 Extension to Functions of Two Variables

In this section, we extend to two variables the concepts introduced in Sections 4.2 through 4.4.

4.5.1 The 2-D Impulse and Its Sifting Property

The impulse, $\delta(t, z)$, of two continuous variables, t and z, is defined as in Eq. (4.2-8):

$$\delta(t, z) = \begin{cases} \infty & \text{if } t = z = 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.5-1a}$$

and

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \delta(t, z)\, dt\, dz = 1 \tag{4.5-1b}$$

As in the 1-D case, the 2-D impulse exhibits the *sifting property* under integration,

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(t, z)\,\delta(t, z)\, dt\, dz = f(0, 0) \tag{4.5-2}$$

or, more generally for an impulse located at coordinates (t_0, z_0),

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(t, z)\,\delta(t - t_0, z - z_0)\, dt\, dz = f(t_0, z_0) \tag{4.5-3}$$

As before, we see that the sifting property yields the value of the function $f(t, z)$ at the location of the impulse.

For discrete variables x and y, the 2-D discrete impulse is defined as

$$\delta(x, y) = \begin{cases} 1 & \text{if } x = y = 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.5-4}$$

and its sifting property is

$$\sum_{x=-\infty}^{\infty}\sum_{y=-\infty}^{\infty} f(x, y)\,\delta(x, y) = f(0, 0) \tag{4.5-5}$$

where $f(x, y)$ is a function of discrete variables x and y. For an impulse located at coordinates (x_0, y_0) (see Fig. 4.12) the sifting property is

$$\sum_{x=-\infty}^{\infty}\sum_{y=-\infty}^{\infty} f(x, y)\,\delta(x - x_0, y - y_0) = f(x_0, y_0) \tag{4.5-6}$$

As before, the sifting property of a discrete impulse yields the value of the discrete function $f(x, y)$ at the location of the impulse.

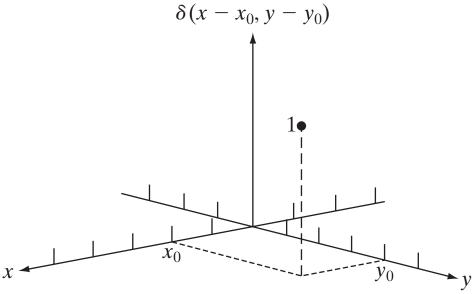


FIGURE 4.12 Two-dimensional unit discrete impulse. Variables x and y are discrete, and δ is zero everywhere except at coordinates (x_0, y_0).

4.5.2 The 2-D Continuous Fourier Transform Pair

Let $f(t, z)$ be a continuous function of two continuous variables, t and z. The two-dimensional, continuous Fourier transform pair is given by the expressions

$$F(\mu, \nu) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(t, z)\, e^{-j2\pi(\mu t + \nu z)}\, dt\, dz \tag{4.5-7}$$

and

$$f(t, z) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(\mu, \nu)\, e^{j2\pi(\mu t + \nu z)}\, d\mu\, d\nu \tag{4.5-8}$$

where μ and ν are the frequency variables. When referring to images, t and z are interpreted to be continuous *spatial* variables. As in the 1-D case, the domain of the variables μ and ν defines the *continuous frequency domain*.

EXAMPLE 4.5: Obtaining the 2-D Fourier transform of a simple function.

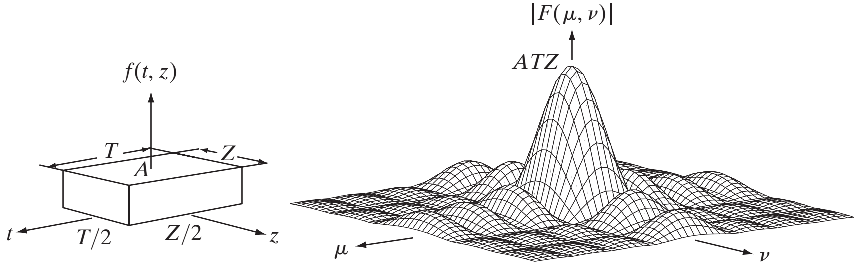
■ Figure 4.13(a) shows a 2-D function analogous to the 1-D case in Example 4.1. Following a procedure similar to the one used in that example gives the result

$$\begin{aligned} F(\mu, \nu) &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(t, z)\, e^{-j2\pi(\mu t + \nu z)}\, dt\, dz \\ &= \int_{-T/2}^{T/2}\int_{-Z/2}^{Z/2} A\, e^{-j2\pi(\mu t + \nu z)}\, dt\, dz \\ &= ATZ\left[\frac{\sin(\pi\mu T)}{(\pi\mu T)}\right]\left[\frac{\sin(\pi\nu Z)}{(\pi\nu Z)}\right] \end{aligned}$$

The magnitude (spectrum) is given by the expression

$$|F(\mu, \nu)| = ATZ\left|\frac{\sin(\pi\mu T)}{(\pi\mu T)}\right|\left|\frac{\sin(\pi\nu Z)}{(\pi\nu Z)}\right|$$

Figure 4.13(b) shows a portion of the spectrum about the origin. As in the 1-D case, the locations of the zeros in the spectrum are inversely proportional to



a b

FIGURE 4.13 (a) A 2-D function, and (b) a section of its spectrum (not to scale). The block is longer along the t-axis, so the spectrum is more "contracted" along the μ-axis. Compare with Fig. 4.4.

the values of T and Z. Thus, the larger T and Z are, the more "contracted" the spectrum will become, and vice versa. ■

4.5.3 Two-Dimensional Sampling and the 2-D Sampling Theorem

In a manner similar to the 1-D case, sampling in two dimensions can be modeled using the sampling function (2-D impulse train):

$$s_{\Delta T \Delta Z}(t, z) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \delta(t - m\Delta T, z - n\Delta Z) \tag{4.5-9}$$

where ΔT and ΔZ are the separations between samples along the t- and z-axis of the continuous function $f(t, z)$. Equation (4.5-9) describes a set of periodic impulses extending infinitely along the two axes (Fig. 4.14). As in the 1-D case illustrated in Fig. 4.5, multiplying $f(t, z)$ by $s_{\Delta T \Delta Z}(t, z)$ yields the sampled function.

Function $f(t, z)$ is said to be *band-limited* if its Fourier transform is 0 outside a rectangle established by the intervals $[-\mu_{\max}, \mu_{\max}]$ and $[-\nu_{\max}, \nu_{\max}]$; that is,

$$F(\mu, \nu) = 0 \quad \text{for } |\mu| \geq \mu_{\max} \text{ and } |\nu| \geq \nu_{\max} \tag{4.5-10}$$

The *two-dimensional sampling theorem* states that a continuous, band-limited function $f(t, z)$ can be recovered with no error from a set of its samples if the sampling intervals are

$$\Delta T < \frac{1}{2\mu_{\max}} \tag{4.5-11}$$

and

$$\Delta Z < \frac{1}{2\nu_{\max}} \tag{4.5-12}$$

or, expressed in terms of the sampling rate, if

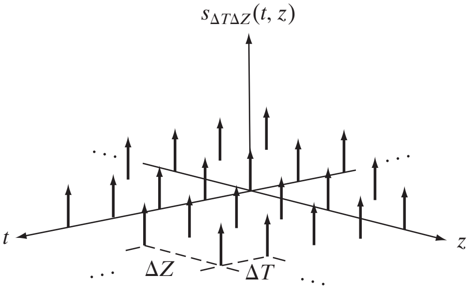


FIGURE 4.14 Two-dimensional impulse train.

$$\frac{1}{\Delta T} > 2\mu_{\max} \tag{4.5-13}$$

and

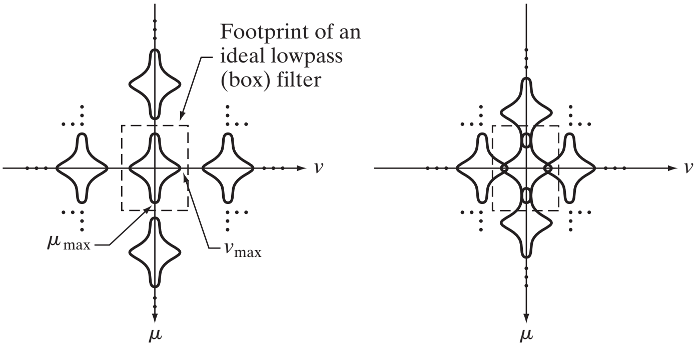
$$\frac{1}{\Delta Z} > 2\nu_{\max} \tag{4.5-14}$$

Stated another way, we say that no information is lost if a 2-D, band-limited, continuous function is represented by samples acquired at rates greater than twice the highest frequency content of the function in both the μ- and ν-directions.

Figure 4.15 shows the 2-D equivalents of Figs. 4.6(b) and (d). A 2-D ideal box filter has the form illustrated in Fig. 4.13(a). The dashed portion of Fig. 4.15(a) shows the location of the filter to achieve the necessary isolation of a single period of the transform for reconstruction of a band-limited function from its samples, as in Section 4.3.3. From Section 4.3.4, we know that if the function is under-sampled the periods overlap, and it becomes impossible to isolate a single period, as Fig. 4.15(b) shows. Aliasing would result under such conditions.

4.5.4 Aliasing in Images

In this section, we extend the concept of aliasing to images and discuss several aspects related to image sampling and resampling.



a b

FIGURE 4.15 Two-dimensional Fourier transforms of (a) an over-sampled, and (b) under-sampled band-limited function.

Extension from 1-D aliasing

As in the 1-D case, a continuous function $f(t, z)$ of two continuous variables, t and z, can be band-limited in general only if it extends infinitely in both coordinate directions. The very act of limiting the duration of the function introduces corrupting frequency components extending to infinity in the frequency domain, as explained in Section 4.3.4. Because we cannot sample a function infinitely, aliasing is always present in digital images, just as it is present in sampled 1-D functions. There are two principal manifestations of aliasing in images: spatial aliasing and temporal aliasing. *Spatial aliasing* is due to under-sampling, as discussed in Section 4.3.4. *Temporal aliasing* is related to time intervals between images in a sequence of images. One of the most common examples of temporal aliasing is the "wagon wheel" effect, in which wheels with spokes in a sequence of images (for example, in a movie) appear to be rotating backwards. This is caused by the frame rate being too low with respect to the speed of wheel rotation in the sequence.

Our focus in this chapter is on spatial aliasing. The key concerns with spatial aliasing in images are the introduction of artifacts such as jaggedness in line features, spurious highlights, and the appearance of frequency patterns not present in the original image. The following example illustrates aliasing in images.

EXAMPLE 4.6:
Aliasing in images.

This example should not be construed as being unrealistic. Sampling a "perfect" scene under noiseless, distortion-free conditions is common when converting computer-generated models and vector drawings to digital images.

■ Suppose that we have an imaging system that is perfect, in the sense that it is noiseless and produces an exact digital image of what it sees, but the number of samples it can take is fixed at 96 × 96 pixels. If we use this system to digitize checkerboard patterns, it will be able to resolve patterns that are up to 96 × 96 squares, in which the size of each square is 1 × 1 pixels. In this limiting case, each pixel in the resulting image will correspond to one square in the pattern. We are interested in examining what happens when the detail (the size of the checkerboard squares) is less than one camera pixel; that is, when the imaging system is asked to digitize checkerboard patterns that have more than 96 × 96 squares in the field of view.

Figures 4.16(a) and (b) show the result of sampling checkerboards whose squares are of size 16 and 6 pixels on the side, respectively. These results are as expected. However, when the size of the squares is reduced to slightly less than one camera pixel a severely aliased image results, as Fig. 4.16(c) shows. Finally, reducing the size of the squares to slightly less than 0.5 pixels on the side yielded the image in Fig. 4.16(d). In this case, the aliased result looks like a normal checkerboard pattern. In fact, this image would result from sampling a checkerboard image whose squares were 12 pixels on the side. This last image is a good reminder that aliasing can create results that may be quite misleading. ■

The effects of aliasing can be *reduced* by slightly defocusing the scene to be digitized so that high frequencies are attenuated. As explained in Section 4.3.4, anti-aliasing filtering has to be done at the "front-end," *before* the image is sampled. There are no such things as after-the-fact software anti-aliasing filters that can be used to reduce the effects of aliasing caused by violations of the sampling theorem. Most commercial digital image manipulation packages do have a feature called "anti-aliasing." However, as illustrated in Examples 4.7

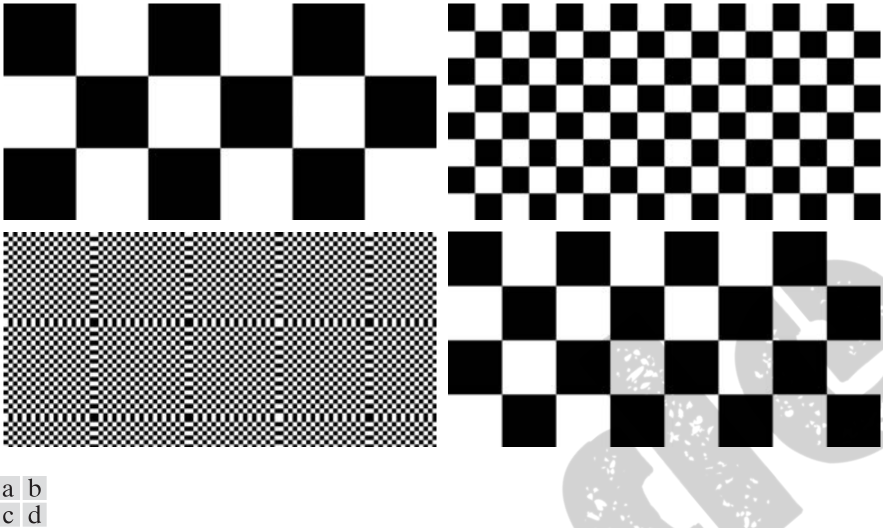


FIGURE 4.16 Aliasing in images. In (a) and (b), the lengths of the sides of the squares are 16 and 6 pixels, respectively, and aliasing is visually negligible. In (c) and (d), the sides of the squares are 0.9174 and 0.4798 pixels, respectively, and the results show significant aliasing. Note that (d) masquerades as a "normal" image.

and 4.8, this term is related to blurring a *digital* image to reduce additional aliasing artifacts caused by resampling. The term does not apply to reducing aliasing in the original sampled image. A significant number of commercial digital cameras have true anti-aliasing filtering built in, either in the lens or on the surface of the sensor itself. For this reason, it is difficult to illustrate aliasing using images obtained with such cameras.

Image interpolation and resampling

As in the 1-D case, perfect reconstruction of a band-limited image function from a set of its samples requires 2-D convolution in the spatial domain with a sinc function. As explained in Section 4.3.5, this theoretically perfect reconstruction requires interpolation using infinite summations which, in practice, forces us to look for approximations. One of the most common applications of 2-D interpolation in image processing is in image resizing (zooming and shrinking). Zooming may be viewed as over-sampling, while shrinking may be viewed as under-sampling. The key difference between these two operations and the sampling concepts discussed in previous sections is that zooming and shrinking are applied to *digital* images.

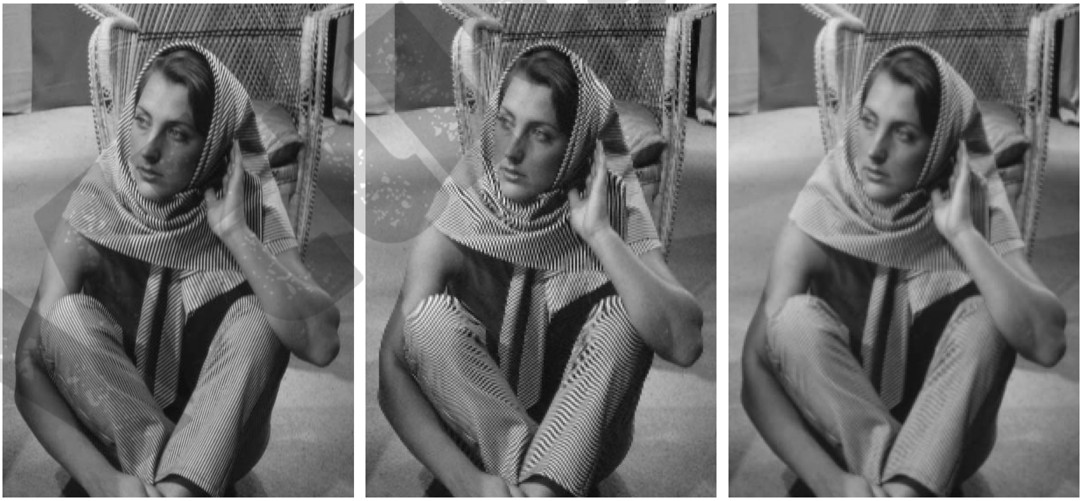
Interpolation was explained in Section 2.4.4. Our interest there was to illustrate the performance of nearest neighbor, bilinear, and bicubic interpolation. In this section, we give some additional examples with a focus on sampling and anti-aliasing issues. A special case of nearest neighbor interpolation that ties in nicely with over-sampling is zooming by *pixel replication*, which is applicable when we want to increase the size of an image an integer number of times. For instance, to double the size of an image, we duplicate each column. This doubles the image size in the horizontal direction. Then, we duplicate each row of the enlarged image to double the size in the vertical direction. The same procedure is used to enlarge the image any integer number of times. The intensity-level assignment of each pixel is predetermined by the fact that new locations are exact duplicates of old locations.

Image shrinking is done in a manner similar to zooming. Under-sampling is achieved by row-column deletion (e.g., to shrink an image by one-half, we delete every other row and column). We can use the zooming grid analogy in Section 2.4.4 to visualize the concept of shrinking by a non-integer factor, except that we now expand the grid to fit over the original image, do intensity-level interpolation, and then shrink the grid back to its specified size. To reduce aliasing, it is a good idea to blur an image slightly before shrinking it (we discuss frequency domain blurring in Section 4.8). An alternate technique is to *super-sample* the original scene and then reduce (resample) its size by row and column deletion. This can yield sharper results than with smoothing, but it clearly requires access to the original scene. Clearly, if we have no access to the original scene (as typically is the case in practice) super-sampling is not an option.

The process of resampling an image without using band-limiting blurring is called *decimation*.

EXAMPLE 4.7: Illustration of aliasing in resampled images.

■ The effects of aliasing generally are worsened when the size of a digital image is reduced. Figure 4.17(a) is an image purposely created to illustrate the effects of aliasing (note the thinly-spaced parallel lines in all garments worn by the subject). There are no objectionable artifacts in Fig. 4.17(a), indicating that



a b c

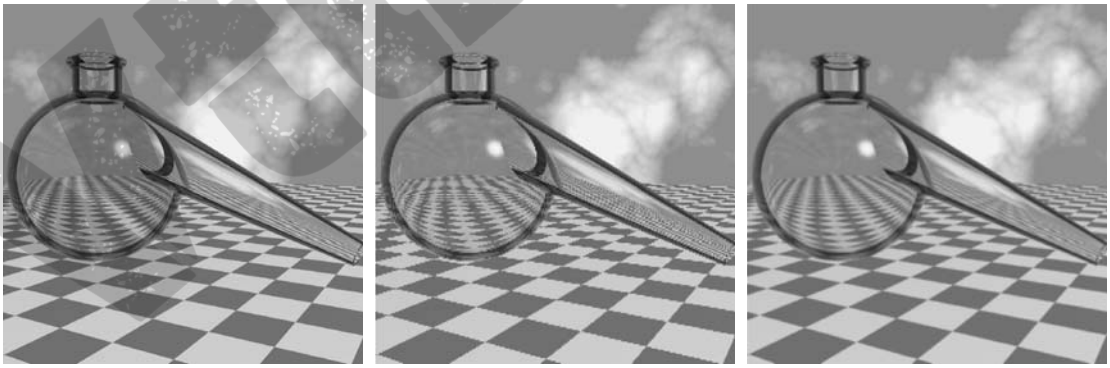
FIGURE 4.17 Illustration of aliasing on resampled images. (a) A digital image with negligible visual aliasing. (b) Result of resizing the image to 50% of its original size by pixel deletion. Aliasing is clearly visible. (c) Result of blurring the image in (a) with a 3 × 3 averaging filter prior to resizing. The image is slightly more blurred than (b), but aliasing is not longer objectionable. (Original image courtesy of the Signal Compression Laboratory, University of California, Santa Barbara.)

the sampling rate used initially was sufficient to avoid visible aliasing. In Fig. 4.17(b), the image was reduced to 50% of its original size using row-column deletion. The effects of aliasing are quite visible in this image (see, for example the areas around the subject's knees). The digital "equivalent" of anti-aliasing filtering of continuous images is to attenuate the high frequencies of a digital image by smoothing it before resampling. Figure 4.17(c) shows the result of smoothing the image in Fig. 4.17(a) with a 3 × 3 averaging filter (see Section 3.5) before reducing its size. The improvement over Fig. 4.17(b) is evident. Images (b) and (c) were resized up to their original dimension by pixel replication to simplify comparisons. ■

When you work with images that have strong edge content, the effects of aliasing are seen as block-like image components, called *jaggies*. The following example illustrates this phenomenon.

EXAMPLE 4.8: Illustration of jaggies in image shrinking.

■ Figure 4.18(a) shows a 1024 × 1024 digital image of a computer-generated scene in which aliasing is negligible. Figure 4.18(b) is the result of reducing the size of (a) by 75% to 256 × 256 pixels using bilinear interpolation and then using pixel replication to bring the image back to its original size in order to make the effects of aliasing (jaggies in this case) more visible. As in Example 4.7, the effects of aliasing can be made less objectionable by smoothing the image before resampling. Figure 4.18(c) is the result of using a 5 × 5 averaging filter prior to reducing the size of the image. As this figure shows, jaggies were reduced significantly. The size reduction and increase to the original size in Fig. 4.18(c) were done using the same approach used to generate Fig. 4.18(b). ■



a b c

FIGURE 4.18 Illustration of jaggies. (a) A 1024 × 1024 digital image of a computer-generated scene with negligible visible aliasing. (b) Result of reducing (a) to 25% of its original size using bilinear interpolation. (c) Result of blurring the image in (a) with a 5 × 5 averaging filter prior to resizing it to 25% using bilinear interpolation. (Original image courtesy of D. P. Mitchell, Mental Landscape, LLC.)

EXAMPLE 4.9: Illustration of jaggies in image zooming.

■ In the previous two examples, we used pixel replication to zoom the small resampled images. This is not a preferred approach in general, as Fig. 4.19 illustrates. Figure 4.19(a) shows a 1024 × 1024 zoomed image generated by pixel replication from a 256 × 256 section out of the center of the image in Fig. 4.18(a). Note the "blocky" edges. The zoomed image in Fig. 4.19(b) was generated from the same 256 × 256 section, but using bilinear interpolation. The edges in this result are considerably smoother. For example, the edges of the bottle neck and the large checkerboard squares are not nearly as blocky in (b) as they are in (a). ■

Moiré patterns

Before leaving this section, we examine another type of artifact, called *moiré patterns*,[†] that sometimes result from sampling scenes with periodic or nearly periodic components. In optics, moiré patterns refer to beat patterns produced between two gratings of approximately equal spacing. These patterns are a common everyday occurrence. We see them, for example, in overlapping insect window screens and on the interference between TV raster lines and striped materials. In digital image processing, the problem arises routinely when scanning media print, such as newspapers and magazines, or in images with periodic components whose spacing is comparable to the spacing between samples. It is important to note that moiré patterns are more general than sampling artifacts. For instance, Fig. 4.20 shows the moiré effect using ink drawings that have not been digitized. Separately, the patterns are clean and void of interference. However, superimposing one pattern on the other creates

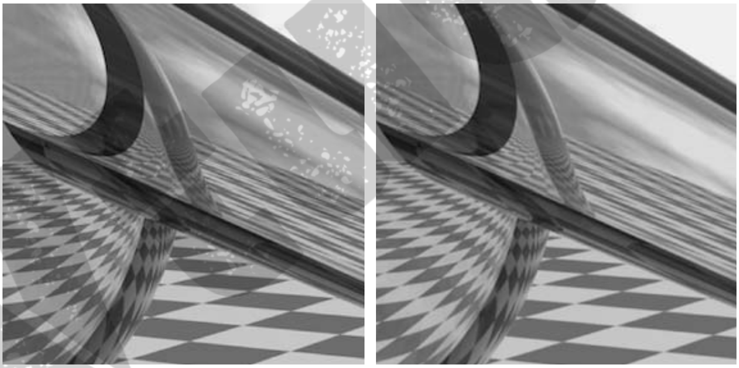
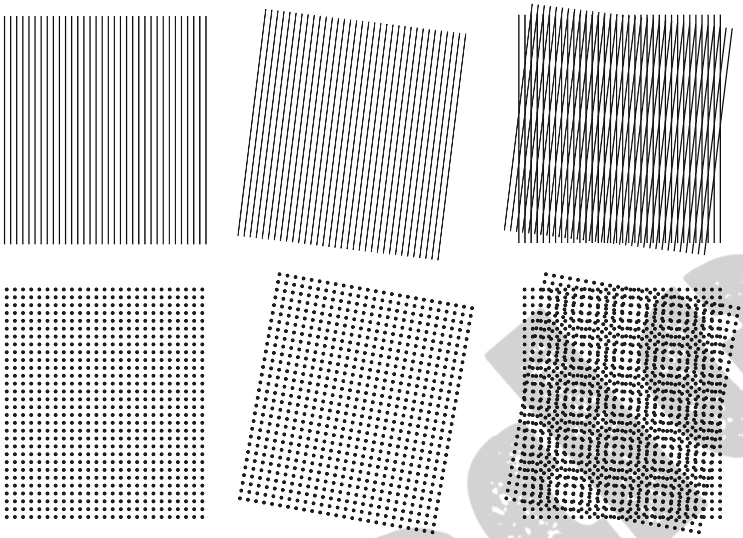


FIGURE 4.19 Image zooming. (a) A 1024 × 1024 digital image generated by pixel replication from a 256 × 256 image extracted from the middle of Fig. 4.18(a). (b) Image generated using bi-linear interpolation, showing a significant reduction in jaggies.

[†] The term *moiré* is a French word (not the name of a person) that appears to have originated with weavers who first noticed interference patterns visible on some fabrics; the term is rooted on the word *mohair*, a cloth made from Angola goat hairs.



a b c
d e f

FIGURE 4.20 Examples of the moiré effect. These are ink drawings, not digitized patterns. Superimposing one pattern on the other is equivalent mathematically to multiplying the patterns.

a beat pattern that has frequencies not present in either of the original patterns. Note in particular the moiré effect produced by two patterns of dots, as this is the effect of interest in the following discussion.

Color printing uses red, green, and blue dots to produce the sensation in the eye of continuous color.

Newspapers and other printed materials make use of so called *halftone dots*, which are black dots or ellipses whose sizes and various joining schemes are used to simulate gray tones. As a rule, the following numbers are typical: newspapers are printed using 75 halftone dots per inch (*dpi* for short), magazines use 133 dpi, and high-quality brochures use 175 dpi. Figure 4.21 shows

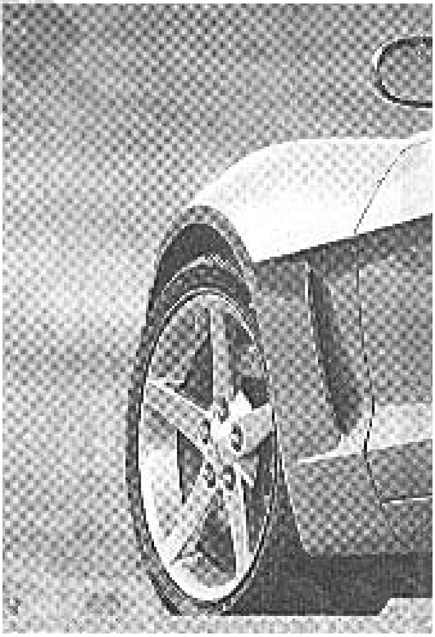


FIGURE 4.21 A newspaper image of size 246 × 168 pixels sampled at 75 dpi showing a moiré pattern. The moiré pattern in this image is the interference pattern created between the ±45° orientation of the halftone dots and the north–south orientation of the sampling grid used to digitize the image.

what happens when a newspaper image is sampled at 75 dpi. The sampling lattice (which is oriented vertically and horizontally) and dot patterns on the newspaper image (oriented at ±45°) interact to create a uniform moiré pattern that makes the image look blotchy. (We discuss a technique in Section 4.10.2 for reducing moiré interference patterns.)

As a related point of interest, Fig. 4.22 shows a newspaper image sampled at 400 dpi to avoid moiré effects. The enlargement of the region surrounding the subject's left eye illustrates how halftone dots are used to create shades of gray. The dot size is inversely proportional to image intensity. In light areas, the dots are small or totally absent (see, for example, the white part of the eye). In light gray areas, the dots are larger, as shown below the eye. In darker areas, when dot size exceeds a specified value (typically 50%), dots are allowed to join along two specified directions to form an interconnected mesh (see, for example, the left part of the eye). In some cases the dots join along only one direction, as in the top right area below the eyebrow.

4.5.5 The 2-D Discrete Fourier Transform and Its Inverse

A development similar to the material in Sections 4.3 and 4.4 would yield the following 2-D *discrete Fourier transform* (DFT):

$$F(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y)\, e^{-j2\pi(ux/M + vy/N)} \tag{4.5-15}$$

where $f(x, y)$ is a digital image of size $M \times N$. As in the 1-D case, Eq. (4.5-15) must be evaluated for values of the discrete variables u and v in the ranges $u = 0, 1, 2, \ldots, M-1$ and $v = 0, 1, 2, \ldots, N-1$.[†]

Sometimes you will find in the literature the $1/MN$ constant in front of DFT instead of the IDFT. At times, the constant is expressed as $1/\sqrt{MN}$ and is included in front of the forward and inverse transforms, thus creating a more symmetric pair. Any of these formulations is correct, provided that you are consistent.

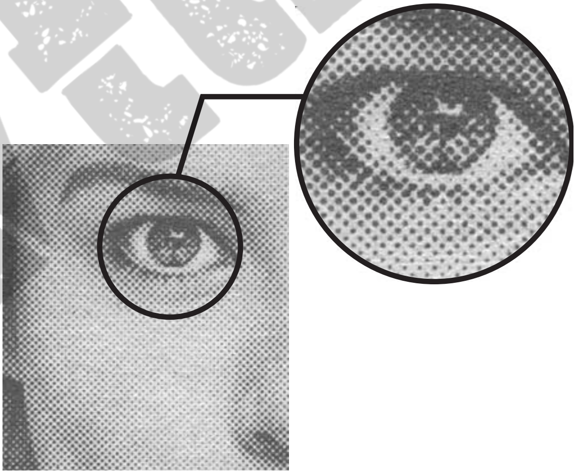


FIGURE 4.22 A newspaper image and an enlargement showing how halftone dots are arranged to render shades of gray.

[†] As mentioned in Section 4.4.1, keep in mind that in this chapter we use (t, z) and (μ, ν) to denote 2-D *continuous* spatial and frequency-domain variables. In the 2-D *discrete* case, we use (x, y) for spatial variables and (u, v) for frequency-domain variables.

Given the transform $F(u, v)$, we can obtain $f(x, y)$ by using the *inverse discrete Fourier transform* (IDFT):

$$f(x, y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u, v) e^{j2\pi(ux/M + vy/N)} \tag{4.5-16}$$

for $x = 0, 1, 2, \ldots, M - 1$ and $y = 0, 1, 2, \ldots, N - 1$. Equations (4.5-15) and (4.5-16) constitute the 2-D *discrete Fourier transform pair*. The rest of this chapter is based on properties of these two equations and their use for image filtering in the frequency domain.

4.6 Some Properties of the 2-D Discrete Fourier Transform

In this section, we introduce several properties of the 2-D discrete Fourier transform and its inverse.

4.6.1 Relationships Between Spatial and Frequency Intervals

The relationships between spatial sampling and the corresponding frequency-domain intervals are as explained in Section 4.4.2. Suppose that a continuous function $f(t, z)$ is sampled to form a digital image, $f(x, y)$, consisting of $M \times N$ samples taken in the t- and z-directions, respectively. Let ΔT and ΔZ denote the separations between samples (see Fig. 4.14). Then, the separations between the corresponding discrete, frequency domain variables are given by

$$\Delta u = \frac{1}{M \Delta T} \tag{4.6-1}$$

and

$$\Delta v = \frac{1}{N \Delta Z} \tag{4.6-2}$$

respectively. Note that the separations between samples in the frequency domain are inversely proportional both to the spacing between spatial samples and the number of samples.

4.6.2 Translation and Rotation

It can be shown by direct substitution into Eqs. (4.5-15) and (4.5-16) that the Fourier transform pair satisfies the following translation properties (Problem 4.16):

$$f(x, y) e^{j2\pi(u_0 x/M + v_0 y/N)} \Leftrightarrow F(u - u_0, v - v_0) \tag{4.6-3}$$

and

$$f(x - x_0, y - y_0) \Leftrightarrow F(u, v) e^{-j2\pi(x_0 u/M + y_0 v/N)} \tag{4.6-4}$$

That is, multiplying $f(x, y)$ by the exponential shown shifts the origin of the DFT to (u_0, v_0) and, conversely, multiplying $F(u, v)$ by the negative of that exponential shifts the origin of $f(x, y)$ to (x_0, y_0). As we illustrate in Example 4.13, translation has no effect on the magnitude (spectrum) of $F(u, v)$.

Using the polar coordinates

$$x = r\cos\theta \quad y = r\sin\theta \quad u = \omega\cos\varphi \quad v = \omega\sin\varphi$$

results in the following transform pair:

$$f(r, \theta + \theta_0) \Leftrightarrow F(\omega, \varphi + \theta_0) \tag{4.6-5}$$

which indicates that rotating $f(x, y)$ by an angle θ_0 rotates $F(u, v)$ by the same angle. Conversely, rotating $F(u, v)$ rotates $f(x, y)$ by the same angle.

4.6.3 Periodicity

As in the 1-D case, the 2-D Fourier transform and its inverse are infinitely periodic in the u and v directions; that is,

$$F(u, v) = F(u + k_1M, v) = F(u, v + k_2N) = F(u + k_1M, v + k_2N) \tag{4.6-6}$$

and

$$f(x, y) = f(x + k_1M, y) = f(x, y + k_2N) = f(x + k_1M, y + k_2N) \tag{4.6-7}$$

where k_1 and k_2 are integers.

The periodicities of the transform and its inverse are important issues in the implementation of DFT-based algorithms. Consider the 1-D spectrum in Fig. 4.23(a). As explained in Section 4.4.1, the transform data in the interval from 0 to $M - 1$ consists of two back-to-back half periods meeting at point $M/2$. For display and filtering purposes, it is more convenient to have in this interval a complete period of the transform in which the data are contiguous, as in Fig. 4.23(b). It follows from Eq. (4.6-3) that

$$f(x)e^{j2\pi(u_0x/M)} \Leftrightarrow F(u - u_0)$$

In other words, multiplying $f(x)$ by the exponential term shown shifts the data so that the origin, $F(0)$, is located at u_0. If we let $u_0 = M/2$, the exponential term becomes $e^{j\pi x}$ which is equal to $(-1)^x$ because x is an integer. In this case,

$$f(x)(-1)^x \Leftrightarrow F(u - M/2)$$

That is, multiplying $f(x)$ by $(-1)^x$ shifts the data so that $F(0)$ is at the *center* of the interval $[0, M - 1]$, which corresponds to Fig. 4.23(b), as desired.

In 2-D the situation is more difficult to graph, but the principle is the same, as Fig. 4.23(c) shows. Instead of two half periods, there are now four quarter periods meeting at the point $(M/2, N/2)$. The dashed rectangles correspond to

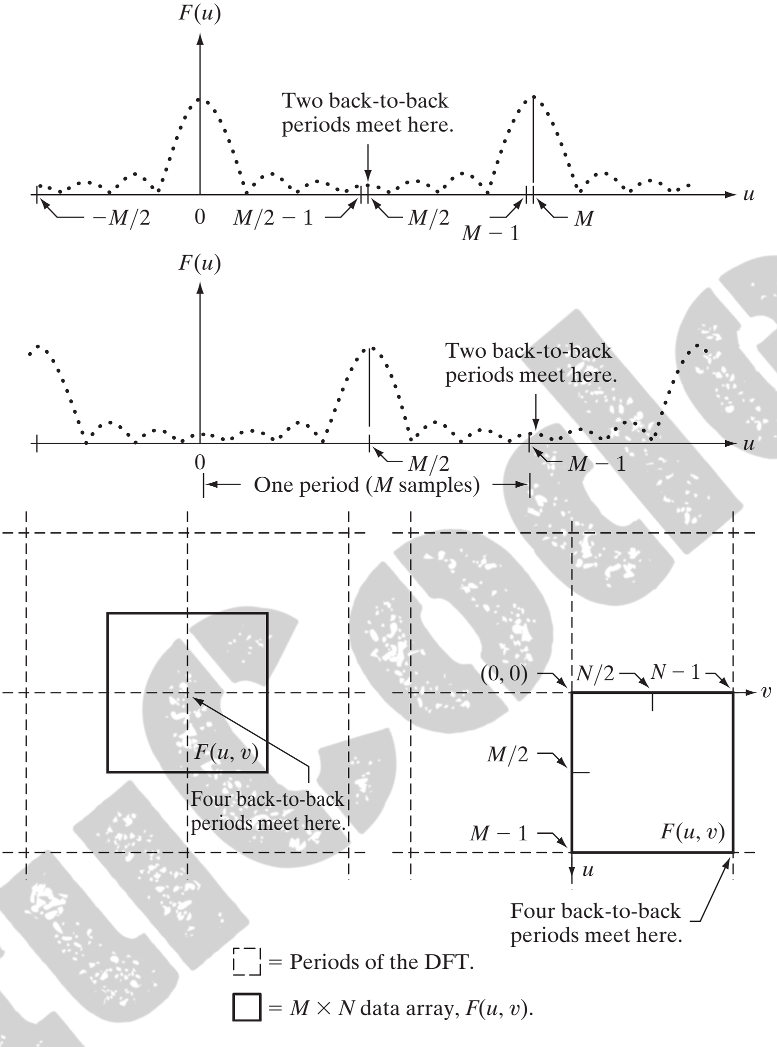

a
b
c d
F(u)
Two back-to-back periods meet here.
u
−M/2
0
M/2 − 1
M/2
M − 1
M
F(u)
Two back-to-back periods meet here.
u
0
M/2
M − 1
One period (M samples)
F(u, v)
Four back-to-back periods meet here.
(0, 0)
N/2
N − 1
v
M/2
M − 1
F(u, v)
u
Four back-to-back periods meet here.
= Periods of the DFT.
= M × N data array, F(u, v).

FIGURE 4.23 Centering the Fourier transform. (a) A 1-D DFT showing an infinite number of periods. (b) Shifted DFT obtained by multiplying $f(x)$ by $(-1)^x$ before computing $F(u)$. (c) A 2-D DFT showing an infinite number of periods. The solid area is the $M \times N$ data array, $F(u, v)$, obtained with Eq. (4.5-15). This array consists of four quarter periods. (d) A Shifted DFT obtained by multiplying $f(x, y)$ by $(-1)^{x+y}$ before computing $F(u, v)$. The data now contains one complete, centered period, as in (b).

the infinite number of periods of the 2-D DFT. As in the 1-D case, visualization is simplified if we shift the data so that $F(0, 0)$ is at $(M/2, N/2)$. Letting $(u_0, v_0) = (M/2, N/2)$ in Eq. (4.6-3) results in the expression

$$f(x, y)(-1)^{x+y} \Leftrightarrow F(u - M/2, v - N/2) \qquad (4.6\text{-}8)$$

Using this equation shifts the data so that $F(0, 0)$ is at the center of the *frequency rectangle* defined by the intervals $[0, M - 1]$ and $[0, N - 1]$, as desired. Figure 4.23(d) shows the result. We illustrate these concepts later in this section as part of Example 4.11 and Fig. 4.24.

4.6.4 Symmetry Properties

An important result from functional analysis is that any real *or* complex function, $w(x, y)$, can be expressed as the sum of an even and an odd part (*each of which can be real or complex*):

$$w(x, y) = w_e(x, y) + w_o(x, y) \tag{4.6-9}$$

where the even and odd parts are defined as

$$w_e(x, y) \triangleq \frac{w(x, y) + w(-x, -y)}{2} \tag{4.6-10a}$$

and

$$w_o(x, y) \triangleq \frac{w(x, y) - w(-x, -y)}{2} \tag{4.6-10b}$$

Substituting Eqs. (4.6-10a) and (4.6-10b) into Eq. (4.6-9) gives the identity $w(x, y) \equiv w(x, y)$, thus proving the validity of the latter equation. It follows from the preceding definitions that

$$w_e(x, y) = w_e(-x, -y) \tag{4.6-11a}$$

and that

$$w_o(x, y) = -w_o(-x, -y) \tag{4.6-11b}$$

Even functions are said to be *symmetric* and odd functions are *antisymmetric*. Because all indices in the DFT and IDFT are positive, when we talk about symmetry (antisymmetry) we are referring to symmetry (antisymmetry) about the *center point* of a sequence. In terms of Eq. (4.6-11), indices to the right of the center point of a 1-D array are considered positive, and those to the left are considered negative (similarly in 2-D). In our work, it is more convenient to think only in terms of nonnegative indices, in which case the definitions of evenness and oddness become:

$$w_e(x, y) = w_e(M - x, N - y) \tag{4.6-12a}$$

and

$$w_o(x, y) = -w_o(M - x, N - y) \tag{4.6-12b}$$

where, as usual, M and N are the number of rows and columns of a 2-D array.

We know from elementary mathematical analysis that the product of two even or two odd functions is even, and that the product of an even and an odd function is odd. In addition, the only way that a discrete function can be odd is if all its samples sum to zero. These properties lead to the important result that

To convince yourself that the samples of an odd function sum to zero, sketch one period of a 1-D sine wave about the origin or any other interval spanning one period.

$$\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} w_e(x, y)\, w_o(x, y) = 0 \tag{4.6-13}$$

for any two discrete even and odd functions w_e and w_o. In other words, because the argument of Eq. (4.6-13) is odd, the result of the summations is 0. The functions can be real or complex.

EXAMPLE 4.10: Even and odd functions.

■ Although evenness and oddness are visualized easily for continuous functions, these concepts are not as intuitive when dealing with discrete sequences. The following illustrations will help clarify the preceding ideas. Consider the 1-D sequence

$$f = \{f(0) \quad f(1) \quad f(2) \quad f(3)\}$$

$$= \{2 \quad 1 \quad 1 \quad 1\}$$

in which $M = 4$. To test for evenness, the condition $f(x) = f(4 - x)$ must be satisfied; that is, we require that

$$f(0) = f(4), \quad f(2) = f(2), \quad f(1) = f(3), \quad f(3) = f(1)$$

Because $f(4)$ is outside the range being examined, and it can be any value, the value of $f(0)$ is immaterial in the test for evenness. We see that the next three conditions are satisfied by the values in the array, so the sequence is even. In fact, we conclude that *any* 4-point even sequence has to have the form

$$\{a \quad b \quad c \quad b\}$$

That is, only the second and last points must be equal in a 4-point even sequence.

An odd sequence has the interesting property that its first term, $w_0(0, 0)$, is always 0, a fact that follows directly from Eq. (4.6-10b). Consider the 1-D sequence

$$g = \{g(0) \quad g(1) \quad g(2) \quad g(3)\}$$

$$= \{0 \quad -1 \quad 0 \quad 1\}$$

We easily can confirm that this is an odd sequence by noting that the terms in the sequence satisfy the condition $g(x) = -g(4 - x)$. For example, $g(1) = -g(3)$. Any 4-point odd sequence has the form

$$\{0 \quad -b \quad 0 \quad b\}$$

That is, when M is an even number, a 1-D odd sequence has the property that the points at locations 0 and $M/2$ always are zero. When M is odd, the first term still has to be 0, but the remaining terms form pairs with equal value but opposite sign.

The preceding discussion indicates that evenness and oddness of sequences depend also on the length of the sequences. For example, we already showed that the sequence $\{0 \quad -1 \quad 0 \quad 1\}$ is odd. However, the sequence $\{0 \quad -1 \quad 0 \quad 1 \quad 0\}$ is neither odd nor even, although the "basic" structure appears to be odd. This is an important issue in interpreting DFT results. We show later in this section that the DFTs of even and odd functions have some very important characteristics. Thus, it often is the case that understanding when a function is odd or even plays a key role in our ability to interpret image results based on DFTs.

The same basic considerations hold in 2-D. For example, the 6×6 2-D sequence

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & -2 & 0 & 2 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

As an exercise, you should use Eq. (4.6-12b) to convince yourself that this 2-D sequence is odd.

is odd. However, adding another row and column of 0s would give a result that is neither odd nor even. Note that the inner structure of this array is a Sobel mask, as discussed in Section 3.6.4. We return to this mask in Example 4.15. ■

Armed with the preceding concepts, we can establish a number of important symmetry properties of the DFT and its inverse. A property used frequently is that the Fourier transform of a *real* function, $f(x, y)$, is *conjugate symmetric*:

$$F^*(u, v) = F(-u, -v) \tag{4.6-14}$$

If $f(x, y)$ is *imaginary*, its Fourier transform is *conjugate antisymmetric*: $F^*(-u, -v) = -F(u, v)$. The proof of Eq. (4.6-14) is as follows:

Conjugate symmetry also is called *hermitian symmetry*. The term *antihermitian* is used sometimes to refer to conjugate antisymmetry.

$$F^*(u, v) = \left[\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y)\, e^{-j2\pi(ux/M + vy/N)} \right]^*$$

$$= \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f^*(x, y) e^{j2\pi(ux/M + vy/N)}$$

$$= \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-j2\pi([-u]x/M + [-v]y/N)}$$

$$= F(-u, -v)$$

where the third step follows from the fact that $f(x, y)$ is real. A similar approach can be used to prove the conjugate antisymmetry exhibited by the transform of imaginary functions.

Table 4.1 lists symmetries and related properties of the DFT that are useful in digital image processing. Recall that the double arrows indicate Fourier transform pairs; that is, for any row in the table, the properties on the right are satisfied by the Fourier transform of the function having the properties listed on the left, and vice versa. For example, entry 5 reads: The DFT of a real function $f(x, y)$, in which (x, y) is replaced by $(-x, -y)$, is $F^*(u, v)$, where $F(u, v)$, the DFT of $f(x, y)$, is a complex function, and vice versa.

TABLE 4.1 Some symmetry properties of the 2-D DFT and its inverse. $R(u, v)$ and $I(u, v)$ are the real and imaginary parts of $F(u, v)$, respectively. The term *complex* indicates that a function has nonzero real and imaginary parts.

	Spatial Domain†		**Frequency Domain**†
1)	$f(x, y)$ real	⇔	$F^*(u, v) = F(-u, -v)$
2)	$f(x, y)$ imaginary	⇔	$F^*(-u, -v) = -F(u, v)$
3)	$f(x, y)$ real	⇔	$R(u, v)$ even; $I(u, v)$ odd
4)	$f(x, y)$ imaginary	⇔	$R(u, v)$ odd; $I(u, v)$ even
5)	$f(-x, -y)$ real	⇔	$F^*(u, v)$ complex
6)	$f(-x, -y)$ complex	⇔	$F(-u, -v)$ complex
7)	$f^*(x, y)$ complex	⇔	$F^*(-u - v)$ complex
8)	$f(x, y)$ real and even	⇔	$F(u, v)$ real and even
9)	$f(x, y)$ real and odd	⇔	$F(u, v)$ imaginary and odd
10)	$f(x, y)$ imaginary and even	⇔	$F(u, v)$ imaginary and even
11)	$f(x, y)$ imaginary and odd	⇔	$F(u, v)$ real and odd
12)	$f(x, y)$ complex and even	⇔	$F(u, v)$ complex and even
13)	$f(x, y)$ complex and odd	⇔	$F(u, v)$ complex and odd

†Recall that x, y, u, and v are *discrete* (integer) variables, with x and u in the range $[0, M - 1]$, and y, and v in the range $[0, N - 1]$. To say that a complex function is *even* means that its real *and* imaginary parts are even, and similarly for an odd complex function.

EXAMPLE 4.11: 1-D illustrations of properties from Table 4.1.

■ With reference to the even and odd concepts discussed earlier and illustrated in Example 4.10, the following 1-D sequences and their transforms are short examples of the properties listed in Table 4.1. The numbers in parentheses on the right are the individual elements of $F(u)$, and similarly for $f(x)$ in the last two properties.

Property	$f(x)$		$F(u)$
3	$\{1\ \ 2\ \ 3\ \ 4\}$	$\Leftrightarrow$	$\{(10)\ (-2+2j)\ (-2)\ (-2-2j)\}$
4	$j\{1\ \ 2\ \ 3\ \ 4\}$	$\Leftrightarrow$	$\{(2.5j)\ (.5-.5j)\ (-.5j)\ (-.5-.5j)\}$
8	$\{2\ \ 1\ \ 1\ \ 1\}$	$\Leftrightarrow$	$\{(5)\ (1)\ (1)\ (1)\}$
9	$\{0\ \ -1\ \ 0\ \ 1\}$	$\Leftrightarrow$	$\{(0)\ (2j)\ (0)\ (-2j)\}$
10	$j\{2\ \ 1\ \ 1\ \ 1\}$	$\Leftrightarrow$	$\{(5j)\ (j)\ (j)\ (j)\}$
11	$j\{0\ \ -1\ \ 0\ \ 1\}$	$\Leftrightarrow$	$\{(0)\ (-2)\ (0)\ (2)\}$
12	$\{(4+4j)\ (3+2j)\ (0+2j)\ (3+2j)\}$	$\Leftrightarrow$	$\{(10+10j)\ (4+2j)\ (-2+2j)\ (4+2j)\}$
13	$\{(0+0j)\ (1+1j)\ (0+0j)\ (-1-j)\}$	$\Leftrightarrow$	$\{(0+0j)\ (2-2j)\ (0+0j)\ (-2+2j)\}$

For example, in property 3 we see that a real function with elements $\{1\ \ 2\ \ 3\ \ 4\}$ has Fourier transform whose real part, $\{10\ \ -2\ \ -2\ \ -2\}$, is even and whose imaginary part, $\{0\ \ 2\ \ 0\ \ -2\}$, is odd. Property 8 tells us that a real even function has a transform that is real and even also. Property 12 shows that an even complex function has a transform that is also complex and even. The other property examples are analyzed in a similar manner. ■

EXAMPLE 4.12: Proving several symmetry properties of the DFT from Table 4.1.

■ In this example, we prove several of the properties in Table 4.1 to develop familiarity with manipulating these important properties, and to establish a basis for solving some of the problems at the end of the chapter. We prove only the properties on the right given the properties on the left. The converse is proved in a manner similar to the proofs we give here.

Consider property 3, which reads: If $f(x, y)$ is a real function, the real part of its DFT is even and the odd part is odd; similarly, if a DFT has real and imaginary parts that are even and odd, respectively, then its IDFT is a real function. We prove this property formally as follows. $F(u, v)$ is complex in general, so it can be expressed as the sum of a real and an imaginary part: $F(u, v) = R(u, v) + jI(u, v)$. Then, $F^*(u, v) = R(u, v) - jI(u, v)$. Also, $F(-u, -v) = R(-u, -v) + jI(-u, -v)$. But, as proved earlier, if $f(x, y)$ is real then $F^*(u, v) = F(-u, -v)$, which, based on the preceding two equations, means that $R(u, v) = R(-u, -v)$ and $I(u, v) = -I(-u, -v)$. In view of Eqs. (4.6-11a) and (4.6-11b), this proves that R is an even function and I is an odd function.

Next, we prove property 8. If $f(x, y)$ is real we know from property 3 that the real part of $F(u, v)$ is even, so to prove property 8 all we have to do is show that if $f(x, y)$ is real *and* even then the imaginary part of $F(u, v)$ is 0 (i.e., F is real). The steps are as follows:

$$F(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y)\, e^{-j2\pi(ux/M + vy/N)}$$

which we can write as

$$
\begin{aligned}
F(u, v) &= \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [f_r(x, y)] e^{-j2\pi(ux/M + vy/N)} \\
&= \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [f_r(x, y)] e^{-j2\pi(ux/M)} e^{-j2\pi(vy/N)} \\
&= \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [\text{even}][\text{even} - j\text{odd}][\text{even} - j\text{odd}] \\
&= \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [\text{even}][\text{even} \cdot \text{even} - 2j\text{even} \cdot \text{odd} - \text{odd} \cdot \text{odd}] \\
&= \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [\text{even} \cdot \text{even}] - 2j \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [\text{even} \cdot \text{odd}] \\
&\qquad\qquad - \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [\text{even} \cdot \text{even}] \\
&= \text{real}
\end{aligned}
$$

The fourth step follows from Euler's equation and the fact that the cos and sin are even and odd functions, respectively. We also know from property 8 that, in addition to being real, f is an even function. The only term in the penultimate line containing imaginary components is the second term, which is 0 according to Eq. (4.6-14). Thus, if f is real and even then F is real. As noted earlier, F is also even because f is real. This concludes the proof.

Finally, we prove the validity of property 6. From the definition of the DFT,

Note that we are not making a change of variable here. We are evaluating the DFT of $f(-x, -y)$, so we simply insert this function into the equation, as we would any other function.

$$
\Im\{f(-x, -y)\} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(-x, -y) e^{-j2\pi(ux/M + vy/N)}
$$

Because of periodicity, $f(-x, -y) = f(M - x, N - y)$. If we now define $m = M - x$ and $n = N - y$, then

$$
\Im\{f(-x, -y)\} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m, n) e^{-j2\pi(u[M-m]/M + v[N-n]/N)}
$$

(To convince yourself that the summations are correct, try a 1-D transform and expand a few terms by hand.) Because $\exp[-j2\pi(\text{integer})] = 1$, it follows that

$$\begin{aligned}\Im\{f(-x,-y)\} &= \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)e^{j2\pi(um/M+vn/N)} \\ &= F(-u,-v)\end{aligned}$$

This concludes the proof. ■

4.6.5 Fourier Spectrum and Phase Angle

Because the 2-D DFT is complex in general, it can be expressed in polar form:

$$F(u,v) = |F(u,v)|e^{j\phi(u,v)} \tag{4.6-15}$$

where the magnitude

$$|F(u,v)| = \left[R^2(u,v) + I^2(u,v)\right]^{1/2} \tag{4.6-16}$$

is called the *Fourier* (or *frequency*) *spectrum*, and

$$\phi(u,v) = \arctan\left[\frac{I(u,v)}{R(u,v)}\right] \tag{4.6-17}$$

is the *phase angle*. Recall from the discussion in Section 4.2.1 that the arctan must be computed using a four-quadrant arctangent, such as MATLAB's `atan2(Imag, Real)` function.

Finally, the *power spectrum* is defined as

$$\begin{aligned}P(u,v) &= |F(u,v)|^2 \\ &= R^2(u,v) + I^2(u,v)\end{aligned} \tag{4.6-18}$$

As before, R and I are the real and imaginary parts of $F(u,v)$ and all computations are carried out for the discrete variables $u = 0, 1, 2, \ldots, M-1$ and $v = 0, 1, 2, \ldots, N-1$. Therefore, $|F(u,v)|$, $\phi(u,v)$, and $P(u,v)$ are arrays of size $M \times N$.

The Fourier transform of a real function is conjugate symmetric [Eq. (4.6-14)], which implies that the spectrum has *even* symmetry about the origin:

$$|F(u,v)| = |F(-u,-v)| \tag{4.6-19}$$

The phase angle exhibits the following *odd* symmetry about the origin:

$$\phi(u,v) = -\phi(-u,-v) \tag{4.6-20}$$

It follows from Eq. (4.5-15) that

$$F(0,0) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)$$

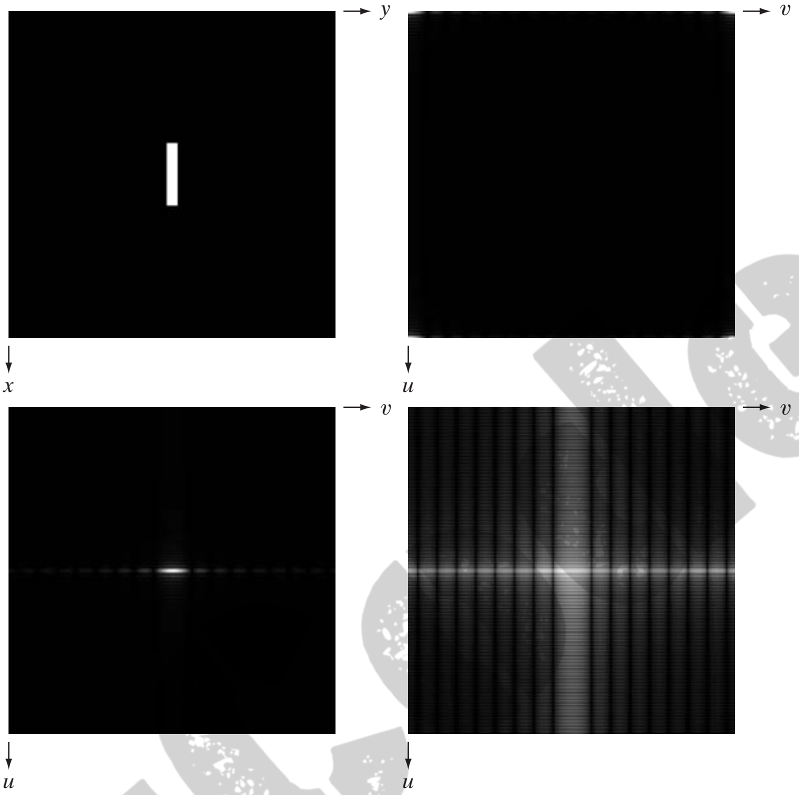


FIGURE 4.24 (a) Image. (b) Spectrum showing bright spots in the four corners. (c) Centered spectrum. (d) Result showing increased detail after a log transformation. The zero crossings of the spectrum are closer in the vertical direction because the rectangle in (a) is longer in that direction. The coordinate convention used throughout the book places the origin of the spatial and frequency domains at the top left.

which indicates that the zero-frequency term is proportional to the average value of $f(x, y)$. That is,

$$\begin{aligned} F(0,0) &= MN\frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y) \\ &= MN\bar{f}(x,y) \qquad (4.6\text{-}21) \end{aligned}$$

where $\bar{f}$ denotes the average value of f. Then,

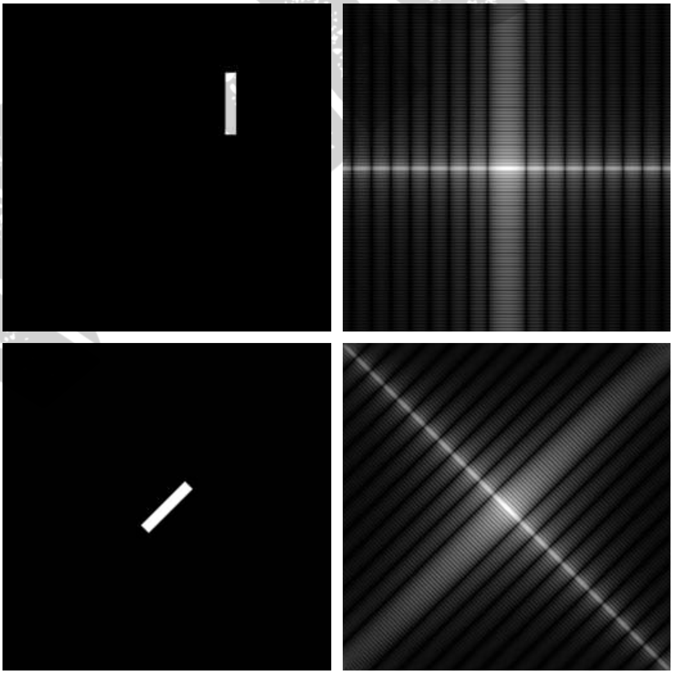
$$|F(0,0)| = MN|\bar{f}(x,y)| \qquad (4.6\text{-}22)$$

Because the proportionality constant MN usually is large, $|F(0,0)|$ typically is the largest component of the spectrum by a factor that can be several orders of magnitude larger than other terms. Because frequency components u and v are zero at the origin, $F(0,0)$ sometimes is called the *dc component* of the transform. This terminology is from electrical engineering, where "dc" signifies direct current (i.e., current of zero frequency).

EXAMPLE 4.13: The 2-D Fourier spectrum of a simple function.

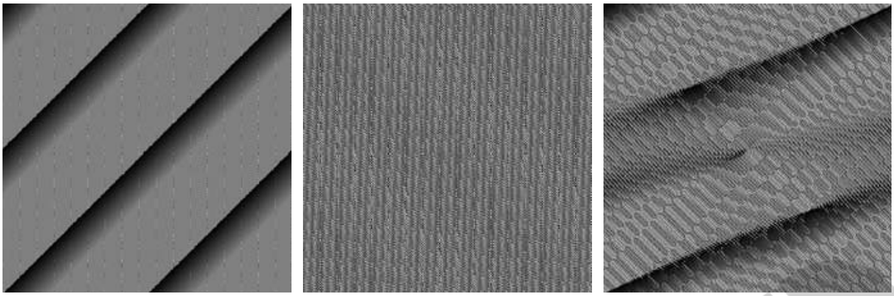
■ Figure 4.24(a) shows a simple image and Fig. 4.24(b) shows its spectrum, whose values were scaled to the range [0, 255] and displayed in image form. The origins of both the spatial and frequency domains are at the top left. Two things are apparent in Fig. 4.22(b). As expected, the area around the origin of the transform contains the highest values (and thus appears brighter in the image). However, note that the four corners of the spectrum contain similarly high values. The reason is the periodicity property discussed in the previous section. To center the spectrum, we simply multiply the image in (a) by $(-1)^{x+y}$ before computing the DFT, as indicated in Eq. (4.6-8). Figure 4.22(c) shows the result, which clearly is much easier to visualize (note the symmetry about the center point). Because the dc term dominates the values of the spectrum, the dynamic range of other intensities in the displayed image are compressed. To bring out those details, we perform a log transformation, as described in Section 3.2.2. Figure 4.24(d) shows the display of $(1 + \log|F(u, v)|)$. The increased rendition of detail is evident. Most spectra shown in this and subsequent chapters are scaled in this manner.

It follows from Eqs. (4.6-4) and (4.6-5) that the spectrum is insensitive to image translation (the absolute value of the exponential term is 1), but it rotates by the same angle of a rotated image. Figure 4.25 illustrates these properties. The spectrum in Fig. 4.25(b) is identical to the spectrum in Fig. 4.24(d). Clearly, the images in Figs. 4.24(a) and 4.25(a) are different, so if their Fourier spectra are the same then, based on Eq. (4.6-15), their phase angles must be different. Figure 4.26 confirms this. Figures 4.26(a) and (b) are the phase angle arrays (shown as images) of the DFTs of Figs. 4.24(a) and 4.25(a). Note the lack of similarity between the phase images, in spite of the fact that the only differences between their corresponding images is simple translation. In general, visual analysis of phase angle images yields little intuitive information. For instance, due to its 45° orientation, one would expect intuitively that the phase angle in



a b
c d

FIGURE 4.25 (a) The rectangle in Fig. 4.24(a) translated, and (b) the corresponding spectrum. (c) Rotated rectangle, and (d) the corresponding spectrum. The spectrum corresponding to the translated rectangle is identical to the spectrum corresponding to the original image in Fig. 4.24(a).



a b c

FIGURE 4.26 Phase angle array corresponding (a) to the image of the centered rectangle in Fig. 4.24(a), (b) to the translated image in Fig. 4.25(a), and (c) to the rotated image in Fig. 4.25(c).

Fig. 4.26(a) should correspond to the rotated image in Fig. 4.25(c), rather than to the image in Fig. 4.24(a). In fact, as Fig. 4.26(c) shows, the phase angle of the rotated image has a strong orientation that is much less than 45°. ■

The components of the spectrum of the DFT determine the amplitudes of the sinusoids that combine to form the resulting image. At any given frequency in the DFT of an image, a large amplitude implies a greater prominence of a sinusoid of that frequency in the image. Conversely, a small amplitude implies that less of that sinusoid is present in the image. Although, as Fig. 4.26 shows, the contribution of the phase components is less intuitive, it is just as important. The phase is a measure of displacement of the various sinusoids with respect to their origin. Thus, while the magnitude of the 2-D DFT is an array whose components determine the intensities in the image, the corresponding phase is an array of angles that carry much of the information about where discernable objects are located in the image. The following example clarifies these concepts further.

EXAMPLE 4.14: Further illustration of the properties of the Fourier spectrum and phase angle.

■ Figure 4.27(b) is the phase angle of the DFT of Fig. 4.27(a). There is no detail in this array that would lead us by visual analysis to associate it with features in its corresponding image (not even the symmetry of the phase angle is visible). However, the importance of the phase in determining shape characteristics is evident in Fig. 4.27(c), which was obtained by computing the inverse DFT of Eq. (4.6-15) using only phase information (i.e., with $|F(u, v)| = 1$ in the equation). Although the intensity information has been lost (remember, that information is carried by the spectrum) the key shape features in this image are unmistakably from Fig. 4.27(a).

Figure 4.27(d) was obtained using only the spectrum in Eq. (4.6-15) and computing the inverse DFT. This means setting the exponential term to 1, which in turn implies setting the phase angle to 0. The result is not unexpected. It contains only intensity information, with the dc term being the most dominant. There is no shape information in the image because the phase was set to zero.

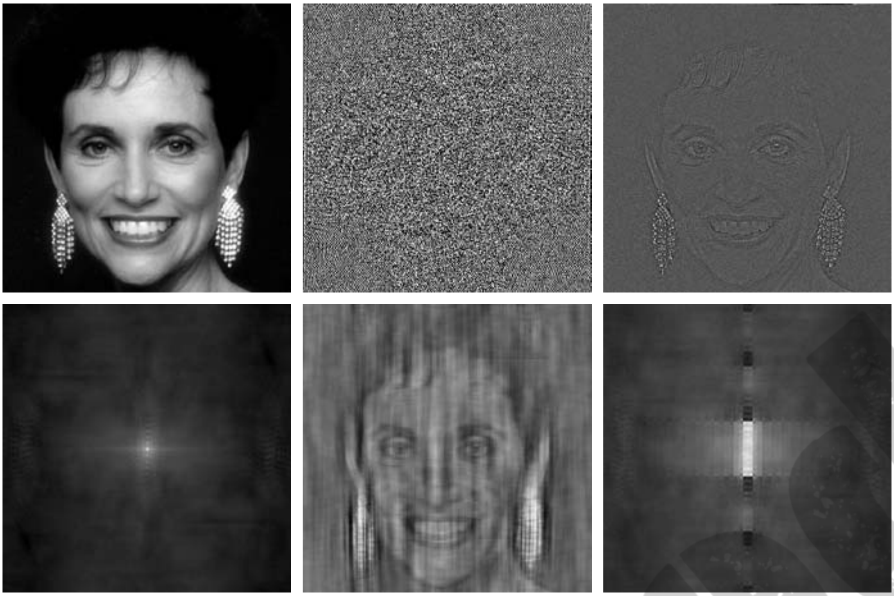


FIGURE 4.27 (a) Woman. (b) Phase angle. (c) Woman reconstructed using only the phase angle. (d) Woman reconstructed using only the spectrum. (e) Reconstruction using the phase angle corresponding to the woman and the spectrum corresponding to the rectangle in Fig. 4.24(a). (f) Reconstruction using the phase of the rectangle and the spectrum of the woman.

Finally, Figs. 4.27(e) and (f) show yet again the dominance of the phase in determining the feature content of an image. Figure 4.27(e) was obtained by computing the IDFT of Eq. (4.6-15) using the spectrum of the rectangle in Fig. 4.24(a) and the phase angle corresponding to the woman. The shape of the woman clearly dominates this result. Conversely, the rectangle dominates Fig. 4.27(f), which was computed using the spectrum of the woman and the phase angle of the rectangle. ■

4.6.6 The 2-D Convolution Theorem

Extending Eq. (4.4-10) to two variables results in the following expression for 2-D *circular convolution*:

$$f(x, y) \star h(x, y) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m, n)h(x - m, y - n) \qquad (4.6\text{-}23)$$

for $x = 0, 1, 2, \ldots, M - 1$ and $y = 0, 1, 2, \ldots, N - 1$. As in Eq. (4.4-10), Eq. (4.6-23) gives one period of a 2-D periodic sequence. The 2-D convolution theorem is given by the expressions

$$f(x, y) \star h(x, y) \Leftrightarrow F(u, v)H(u, v) \qquad (4.6\text{-}24)$$

and, conversely,

$$f(x, y)h(x, y) \Leftrightarrow F(u, v) \star H(u, v) \tag{4.6-25}$$

where F and H are obtained using Eq. (4.5-15) and, as before, the double arrow is used to indicate that the left and right sides of the expressions constitute a Fourier transform pair. Our interest in the remainder of this chapter is in Eq. (4.6-24), which states that the inverse DFT of the product $F(u, v)H(u, v)$ yields $f(x, y) \star h(x, y)$, the 2-D spatial convolution of f and h. Similarly, the DFT of the spatial convolution yields the product of the transforms in the frequency domain. Equation (4.6-24) is the foundation of linear filtering and, as explained in Section 4.7, is the basis for all the filtering techniques discussed in this chapter.

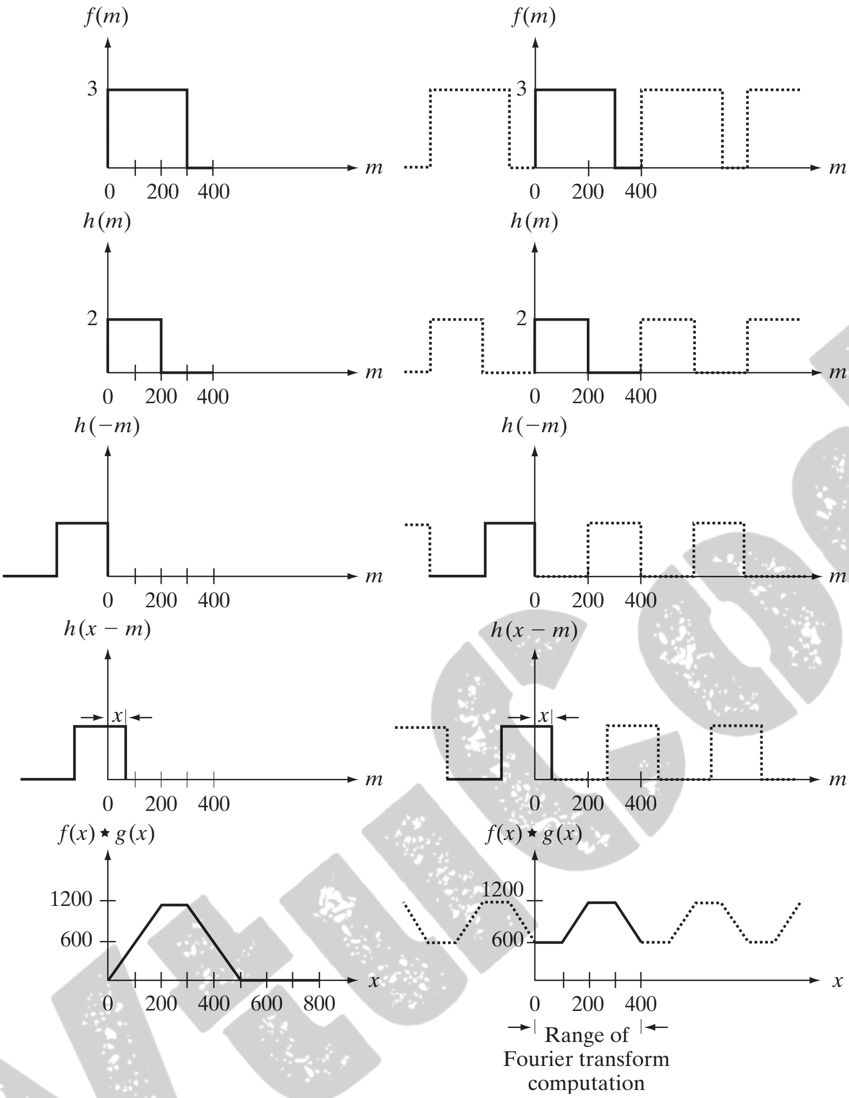
We discuss efficient ways to compute the DFT in Section 4.11.

Because we are dealing here with discrete quantities, computation of the Fourier transforms is carried out with a DFT algorithm. If we elect to compute the spatial convolution using the IDFT of the product of the two transforms, then the periodicity issues discussed in Section 4.6.3 must be taken into account. We give a 1-D example of this and then extend the conclusions to two variables. The left column of Fig. 4.28 implements convolution of two functions, f and h, using the 1-D equivalent of Eq. (3.4-2) which, because the two functions are of same size, is written as

$$f(x) \star h(x) = \sum_{m=0}^{399} f(x)h(x - m)$$

This equation is identical to Eq. (4.4-10), *but* the requirement on the displacement x is that it be sufficiently large to cause the flipped (rotated) version of h to slide completely past f. In other words, the procedure consists of (1) mirroring h about the origin (i.e., rotating it by 180°) [Fig. 4.28(c)], (2) translating the mirrored function by an amount x [Fig. 4.28(d)], and (3) for *each* value x of translation, computing the *entire* sum of products in the right side of the preceding equation. In terms of Fig. 4.28 this means multiplying the function in Fig. 4.28(a) by the function in Fig. 4.28(d) for *each* value of x. The displacement x ranges over all values required to completely slide h across f. Figure 4.28(e) shows the convolution of these two functions. Note that convolution is a function of the displacement variable, x, and that the range of x required in this example to completely slide h past f is from 0 to 799.

If we use the DFT and the convolution theorem to obtain the same result as in the left column of Fig. 4.28, we must take into account the periodicity inherent in the expression for the DFT. This is equivalent to convolving the two periodic functions in Figs. 4.28(f) and (g). The convolution procedure is the same as we just discussed, but the two functions now are periodic. Proceeding with these two functions as in the previous paragraph would yield the result in Fig. 4.28(j) which obviously is incorrect. Because we are convolving two periodic functions, the convolution itself is periodic. The closeness of the periods in Fig. 4.28 is such that they interfere with each other to cause what is commonly referred to as *wraparound error*. According to the convolution theorem, if we had computed the DFT of the two 400-point functions, f and h, multiplied the



a	f
b	g
c	h
d	i
e	j

FIGURE 4.28 Left column: convolution of two discrete functions obtained using the approach discussed in Section 3.4.2. The result in (e) is correct. Right column: Convolution of the same functions, but taking into account the periodicity implied by the DFT. Note in (j) how data from adjacent periods produce wraparound error, yielding an incorrect convolution result. To obtain the correct result, function padding must be used.

two transforms, and then computed the inverse DFT, we would have obtained the erroneous 400-point segment of the convolution shown in Fig. 4.28(j).

Fortunately, the solution to the wraparound error problem is simple. Consider two functions, $f(x)$ and $h(x)$ composed of A and B samples, respectively. It can be shown (Brigham [1988]) that if we append zeros to both functions so that they have the same length, denoted by P, then wraparound is avoided by choosing

$$P \geq A + B - 1 \tag{4.6-26}$$

In our example, each function has 400 points, so the minimum value we could use is $P = 799$, which implies that we would append 399 zeros to the trailing edge of each function. This process is called *zero padding*. As an exercise, you

The zeros could be appended also to the beginning of the functions, or they could be divided between the beginning and end of the functions. It is simpler to append them at the end.

should convince yourself that if the periods of the functions in Figs. 4.28(f) and (g) were lengthened by appending to each period at least 399 zeros, the result would be a periodic convolution in which *each* period is identical to the correct result in Fig. 4.28(e). Using the DFT via the convolution theorem would result in a 799-point spatial function identical to Fig. 4.28(e). The conclusion, then, is that to obtain the same convolution result between the "straight" representation of the convolution equation approach in Chapter 3, and the DFT approach, functions in the latter must be padded prior to computing their transforms.

Visualizing a similar example in 2-D would be more difficult, but we would arrive at the same conclusion regarding wraparound error and the need for appending zeros to the functions. Let $f(x, y)$ and $h(x, y)$ be two image arrays of sizes $A \times B$ and $C \times D$ pixels, respectively. Wraparound error in their circular convolution can be avoided by padding these functions with zeros, as follows:

$$f_p(x, y) = \begin{cases} f(x, y) & 0 \leq x \leq A - 1 \quad \text{and} \quad 0 \leq y \leq B - 1 \\ 0 & A \leq x \leq P \quad \text{or} \quad B \leq y \leq Q \end{cases} \tag{4.6-27}$$

and

$$h_p(x, y) = \begin{cases} h(x, y) & 0 \leq x \leq C - 1 \quad \text{and} \quad 0 \leq y \leq D - 1 \\ 0 & C \leq x \leq P \quad \text{or} \quad D \leq y \leq Q \end{cases} \tag{4.6-28}$$

with

$$P \geq A + C - 1 \tag{4.6-29}$$

and

$$Q \geq B + D - 1 \tag{4.6-30}$$

The resulting padded images are of size $P \times Q$. If both arrays are of the same size, $M \times N$, then we require that

$$P \geq 2M - 1 \tag{4.6-31}$$

and

$$Q \geq 2N - 1 \tag{4.6-32}$$

We give an example in Section 4.7.2 showing the effects of wraparound error on images. As rule, DFT algorithms tend to execute faster with arrays of even size, so it is good practice to select P and Q as the smallest even integers that satisfy the preceding equations. If the two arrays are of the same size, this means that P and Q are selected as twice the array size.

The two functions in Figs. 4.28(a) and (b) conveniently become zero before the end of the sampling interval. If one or both of the functions were not zero at the end of the interval, then a discontinuity would be created when zeros were appended to the function to eliminate wraparound error. This is analogous to multiplying a function by a box, which in the frequency domain would imply convolution of the original transform with a sinc function (see Example 4.1). This, in turn, would create so-called *frequency leakage*, caused by the high-frequency components of the sinc function. Leakage produces a blocky effect on images. Although leakage never can be totally eliminated, it can be reduced significantly by multiplying the sampled function by another function that tapers smoothly to near zero at both ends of the sampled record to dampen the sharp transitions (and thus the high frequency components) of the box. This approach, called *windowing* or *apodizing*, is an important consideration when fidelity in image reconstruction (as in high-definition graphics) is desired. If you are faced with the need for windowing, a good approach is to use a 2-D Gaussian function (see Section 4.8.3). One advantage of this function is that its Fourier transform is Gaussian also, thus producing low leakage.

A simple apodizing function is a triangle, centered on the data record, which tapers to 0 at both ends of the record. This is called the *Bartlett* window. Other common windows are the *Hamming* and the *Hann* windows. We can even use a Gaussian function. We return to the issue of windowing in Section 5.11.5.

4.6.7 Summary of 2-D Discrete Fourier Transform Properties

Table 4.2 summarizes the principal DFT definitions introduced in this chapter. Separability is discussed in Section 4.11.1 and obtaining the inverse using a forward transform algorithm is discussed in Section 4.11.2. Correlation is discussed in Chapter 12.

TABLE 4.2
Summary of DFT definitions and corresponding expressions.

Name	Expression(s)
1) Discrete Fourier transform (DFT) of $f(x, y)$	$F(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-j2\pi(ux/M + vy/N)}$
2) Inverse discrete Fourier transform (IDFT) of $F(u, v)$	$f(x, y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u, v) e^{j2\pi(ux/M + vy/N)}$
3) Polar representation	$F(u, v) = \lvert F(u, v)\rvert e^{j\phi(u,v)}$
4) Spectrum	$\lvert F(u, v)\rvert = \left[R^2(u, v) + I^2(u, v)\right]^{1/2}$ $R = \text{Real}(F); \quad I = \text{Imag}(F)$
5) Phase angle	$\phi(u, v) = \tan^{-1}\left[\frac{I(u, v)}{R(u, v)}\right]$
6) Power spectrum	$P(u, v) = \lvert F(u, v)\rvert^2$
7) Average value	$\bar{f}(x, y) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) = \frac{1}{MN} F(0, 0)$

(*Continued*)

TABLE 4.2
(*Continued*)

Name	Expression(s)
8) Periodicity (k_1 and k_2 are integers)	$F(u,v) = F(u+k_1M, v) = F(u, v+k_2N)$ $= F(u+k_1M, v+k_2N)$ $f(x,y) = f(x+k_1M, y) = f(x, y+k_2N)$ $= f(x+k_1M, y+k_2N)$
9) Convolution	$f(x,y) \bigstar h(x,y) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)h(x-m, y-n)$
10) Correlation	$f(x,y) \star h(x,y) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f^*(m,n)h(x+m, y+n)$
11) Separability	The 2-D DFT can be computed by computing 1-D DFT transforms along the rows (columns) of the image, followed by 1-D transforms along the columns (rows) of the result. See Section 4.11.1.
12) Obtaining the inverse Fourier transform using a forward transform algorithm.	$MNf^*(x,y) = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F^*(u,v)e^{-j2\pi(ux/M+vy/N)}$ This equation indicates that inputting $F^*(u,v)$ into an algorithm that computes the forward transform (right side of above equation) yields $MNf^*(x,y)$. Taking the complex conjugate and dividing by MN gives the desired inverse. See Section 4.11.2.

Table 4.3 summarizes some important DFT pairs. Although our focus is on discrete functions, the last two entries in the table are Fourier transform pairs that can be derived only for continuous variables (note the use of continuous variable notation). We include them here because, with proper interpretation, they are quite useful in digital image processing. The differentiation pair can

TABLE 4.3
Summary of DFT pairs. The closed-form expressions in 12 and 13 are valid only for continuous variables. They can be used with discrete variables by sampling the closed-form, continuous expressions.

Name	DFT Pairs
1) Symmetry properties	See Table 4.1
2) Linearity	$af_1(x,y) + bf_2(x,y) \Leftrightarrow aF_1(u,v) + bF_2(u,v)$
3) Translation (general)	$f(x,y)e^{j2\pi(u_0x/M+v_0y/N)} \Leftrightarrow F(u-u_0, v-v_0)$ $f(x-x_0, y-y_0) \Leftrightarrow F(u,v)e^{-j2\pi(ux_0/M+vy_0/N)}$
4) Translation to center of the frequency rectangle, $(M/2, N/2)$	$f(x,y)(-1)^{x+y} \Leftrightarrow F(u-M/2, v-N/2)$ $f(x-M/2, y-N/2) \Leftrightarrow F(u,v)(-1)^{u+v}$
5) Rotation	$f(r, \theta+\theta_0) \Leftrightarrow F(\omega, \varphi+\theta_0)$ $x = r\cos\theta \quad y = r\sin\theta \quad u = \omega\cos\varphi \quad v = \omega\sin\varphi$
6) Convolution theorem†	$f(x,y) \bigstar h(x,y) \Leftrightarrow F(u,v)H(u,v)$ $f(x,y)h(x,y) \Leftrightarrow F(u,v) \bigstar H(u,v)$

(*Continued*)

TABLE 4.3 (*Continued*)

Name	DFT Pairs
7) Correlation theorem†	$f(x, y) \star h(x, y) \Leftrightarrow F^*(u, v)\,H(u, v)$ $f^*(x, y)h(x, y) \Leftrightarrow F(u, v) \star H(u, v)$
8) Discrete unit impulse	$\delta(x, y) \Leftrightarrow 1$
9) Rectangle	$\text{rect}[a, b] \Leftrightarrow ab\dfrac{\sin(\pi ua)}{(\pi ua)}\dfrac{\sin(\pi vb)}{(\pi vb)}e^{-j\pi(ua+vb)}$
10) Sine	$\sin(2\pi u_0 x + 2\pi v_0 y) \Leftrightarrow$ $j\dfrac{1}{2}\big[\delta(u + Mu_0, v + Nv_0) - \delta(u - Mu_0, v - Nv_0)\big]$
11) Cosine	$\cos(2\pi u_0 x + 2\pi v_0 y) \Leftrightarrow$ $\dfrac{1}{2}\big[\delta(u + Mu_0, v + Nv_0) + \delta(u - Mu_0, v - Nv_0)\big]$
The following Fourier transform pairs are derivable only for continuous variables, denoted as before by t and z for spatial variables and by μ and ν for frequency variables. These results can be used for DFT work by sampling the continuous forms.	
12) *Differentiation* (The expressions on the right assume that $f(\pm\infty, \pm\infty) = 0$.)	$\left(\dfrac{\partial}{\partial t}\right)^m\left(\dfrac{\partial}{\partial z}\right)^n f(t, z) \Leftrightarrow (j2\pi\mu)^m(j2\pi\nu)^n F(\mu, \nu)$ $\dfrac{\partial^m f(t, z)}{\partial t^m} \Leftrightarrow (j2\pi\mu)^m F(\mu, \nu);\ \dfrac{\partial^n f(t, z)}{\partial z^n} \Leftrightarrow (j2\pi\nu)^n F(\mu, \nu)$
13) *Gaussian*	$A2\pi\sigma^2 e^{-2\pi^2\sigma^2(t^2+z^2)} \Leftrightarrow Ae^{-(\mu^2+\nu^2)/2\sigma^2}$ (A is a constant)

†Assumes that the functions have been extended by zero padding. Convolution and correlation are associative, commutative, and distributive.

be used to derive the frequency-domain equivalent of the Laplacian defined in Eq. (3.6-3) (Problem 4.26). The Gaussian pair is discussed in Section 4.7.4.

Tables 4.1 through 4.3 provide a summary of properties useful when working with the DFT. Many of these properties are key elements in the development of the material in the rest of this chapter, and some are used in subsequent chapters.

4.7 The Basics of Filtering in the Frequency Domain

In this section, we lay the groundwork for all the filtering techniques discussed in the remainder of the chapter.

4.7.1 Additional Characteristics of the Frequency Domain

We begin by observing in Eq. (4.5-15) that *each* term of $F(u, v)$ contains *all* values of $f(x, y)$, modified by the values of the exponential terms. Thus, with the exception of trivial cases, it usually is impossible to make direct associations between specific components of an image and its transform. However, some general statements can be made about the relationship between the frequency components of the Fourier transform and spatial features of an image. For instance, because frequency is directly related to spatial rates of change, it is not difficult intuitively to associate frequencies in the Fourier transform with patterns of intensity variations in an image. We showed in Section 4.6.5 that the slowest varying frequency component ($u = v = 0$) is proportional to the average intensity of an image. As we move away from the origin of the transform, the low frequencies correspond to the slowly varying intensity components of an image. In an image of a room, for example, these might correspond to smooth intensity variations on the walls and floor. As we move further away from the origin, the higher frequencies begin to correspond to faster and faster intensity changes in the image. These are the edges of objects and other components of an image characterized by abrupt changes in intensity.

Filtering techniques in the frequency domain are based on modifying the Fourier transform to achieve a specific objective and then computing the inverse DFT to get us back to the image domain, as introduced in Section 2.6.7. It follows from Eq. (4.6-15) that the two components of the transform to which we have access are the transform magnitude (spectrum) and the phase angle. Section 4.6.5 covered the basic properties of these two components of the transform. We learned there that visual analysis of the phase component generally is not very useful. The spectrum, however, provides some useful guidelines as to gross characteristics of the image from which the spectrum was generated. For example, consider Fig. 4.29(a), which is a scanning electron microscope image of an integrated circuit, magnified approximately 2500 times. Aside from the interesting construction of the device itself, we note two principal features: strong edges that run approximately at ±45° and two white, oxide protrusions resulting from thermally-induced failure. The Fourier spectrum in Fig. 4.29(b) shows prominent components along the ±45° directions that correspond to the edges just mentioned. Looking carefully along the vertical axis, we see a vertical component

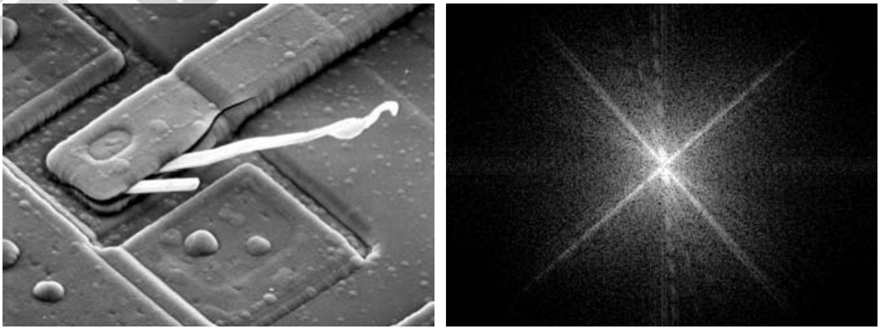


FIGURE 4.29 (a) SEM image of a damaged integrated circuit. (b) Fourier spectrum of (a). (Original image courtesy of Dr. J. M. Hudak, Brockhouse Institute for Materials Research, McMaster University, Hamilton, Ontario, Canada.)

that is off-axis slightly to the left. This component was caused by the edges of the oxide protrusions. Note how the angle of the frequency component with respect to the vertical axis corresponds to the inclination (with respect to the horizontal axis) of the long white element, and note also the zeros in the vertical frequency component, corresponding to the narrow vertical span of the oxide protrusions.

These are typical of the types of associations that can be made in general between the frequency and spatial domains. As we show later in this chapter, even these types of gross associations, coupled with the relationships mentioned previously between frequency content and rate of change of intensity levels in an image, can lead to some very useful results. In the next section, we show the effects of modifying various frequency ranges in the transform of Fig. 4.29(a).

4.7.2 Frequency Domain Filtering Fundamentals

Filtering in the frequency domain consists of modifying the Fourier transform of an image and then computing the inverse transform to obtain the processed result. Thus, given a digital image, $f(x, y)$, of size $M \times N$, the basic filtering equation in which we are interested has the form:

$$g(x, y) = \Im^{-1}[H(u, v)F(u, v)] \tag{4.7-1}$$

where $\Im^{-1}$ is the IDFT, $F(u, v)$ is the DFT of the input image, $f(x, y)$, $H(u, v)$ is a *filter function* (also called simply the *filter*, or the *filter transfer function*), and $g(x, y)$ is the filtered (output) image. Functions F, H, and g are arrays of size $M \times N$, the same as the input image. The product $H(u, v)F(u, v)$ is formed using array multiplication, as defined in Section 2.6.1. The filter function modifies the transform of the input image to yield a processed output, $g(x, y)$. Specification of $H(u, v)$ is simplified considerably by using functions that are symmetric about their center, which requires that $F(u, v)$ be centered also. As explained in Section 4.6.3, this is accomplished by multiplying the input image by $(-1)^{x+y}$ prior to computing its transform.†

If H is real and symmetric and f is real (as is typically the case), then the IDFT in Eq. (4.7-1) should yield real quantities in theory. In practice, the inverse generally contains parasitic complex terms from roundoff and other computational inaccuracies. Thus, it is customary to take the real part of the IDFT to form g.

We are now in a position to consider the filtering process in some detail. One of the simplest filters we can construct is a filter $H(u, v)$ that is 0 at the center of the transform and 1 elsewhere. This filter would reject the dc term and "pass" (i.e., leave unchanged) all other terms of $F(u, v)$ when we form the product $H(u, v)F(u, v)$. We know from Eq. (4.6-21) that the dc term is responsible for the average intensity of an image, so setting it to zero will reduce the average intensity of the output image to zero. Figure 4.30 shows the result of this operation using Eq. (4.7-1). As expected, the image became much darker. (An average of zero

† Many software implementations of the 2-D DFT (e.g., MATLAB) do not center the transform. This implies that filter functions must be arranged to correspond to the same data format as the uncentered transform (i.e., with the origin at the top left). The net result is that filters are more difficult to generate and display. We use centering in our discussions to aid in visualization, which is crucial in developing a clear understanding of filtering concepts. Either method can be used practice, as long as consistency is maintained.

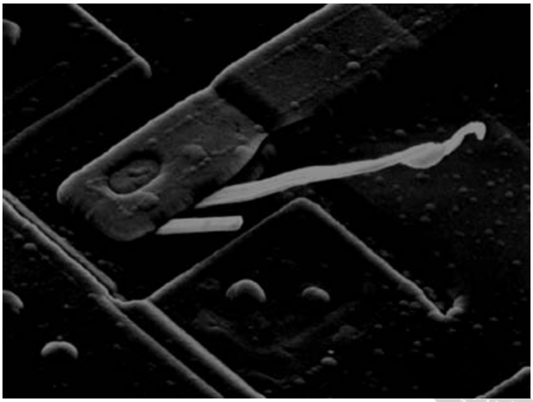


FIGURE 4.30 Result of filtering the image in Fig. 4.29(a) by setting to 0 the term $F(M/2, N/2)$ in the Fourier transform.

implies the existence of negative intensities. Therefore, although it illustrates the principle, Fig. 4.30 is not a true representation of the original, as all negative intensities were clipped (set to 0) for display purposes.)

As noted earlier, low frequencies in the transform are related to slowly varying intensity components in an image, such as the walls of a room or a cloudless sky in an outdoor scene. On the other hand, high frequencies are caused by sharp transitions in intensity, such as edges and noise. Therefore, we would expect that a filter $H(u, v)$ that attenuates high frequencies while passing low frequencies (appropriately called a *lowpass filter*) would blur an image, while a filter with the opposite property (called a *highpass filter*) would enhance sharp detail, but cause a reduction in contrast in the image. Figure 4.31 illustrates these effects. Note the similarity between Figs. 4.31(e) and Fig. 4.30. The reason is that the highpass filter shown eliminates the dc term, resulting in the same basic effect that led to Fig. 4.30. Adding a small constant to the filter does not affect sharpening appreciably, but it does prevent elimination of the dc term and thus preserves tonality, as Fig. 4.31(f) shows.

Equation (4.7-1) involves the product of two functions in the frequency domain which, by the convolution theorem, implies convolution in the spatial domain. We know from the discussion in Section 4.6.6 that if the functions in question are not padded we can expect wraparound error. Consider what happens when we apply Eq. (4.7-1) without padding. Figure 4.32(a) shows a simple image, and Fig. 4.32(b) is the result of lowpass filtering the image with a Gaussian lowpass filter of the form shown in Fig. 4.31(a). As expected, the image is blurred. However, the blurring is not uniform; the top white edge is blurred, but the side white edges are not. Padding the input image according to Eqs. (4.6-31) and (4.6-32) before applying Eq. (4.7-1) results in the filtered image in Fig. 4.32(c). This result is as expected.

Figure 4.33 illustrates the reason for the discrepancy between Figs. 4.32(b) and (c). The dashed areas in Fig. 4.33 correspond to the image in Fig. 4.32(a). Figure 4.33(a) shows the periodicity implicit in the use of the DFT, as explained in Section 4.6.3. Imagine convolving the *spatial* representation of the blurring filter with this image. When the filter is passing through the top of the

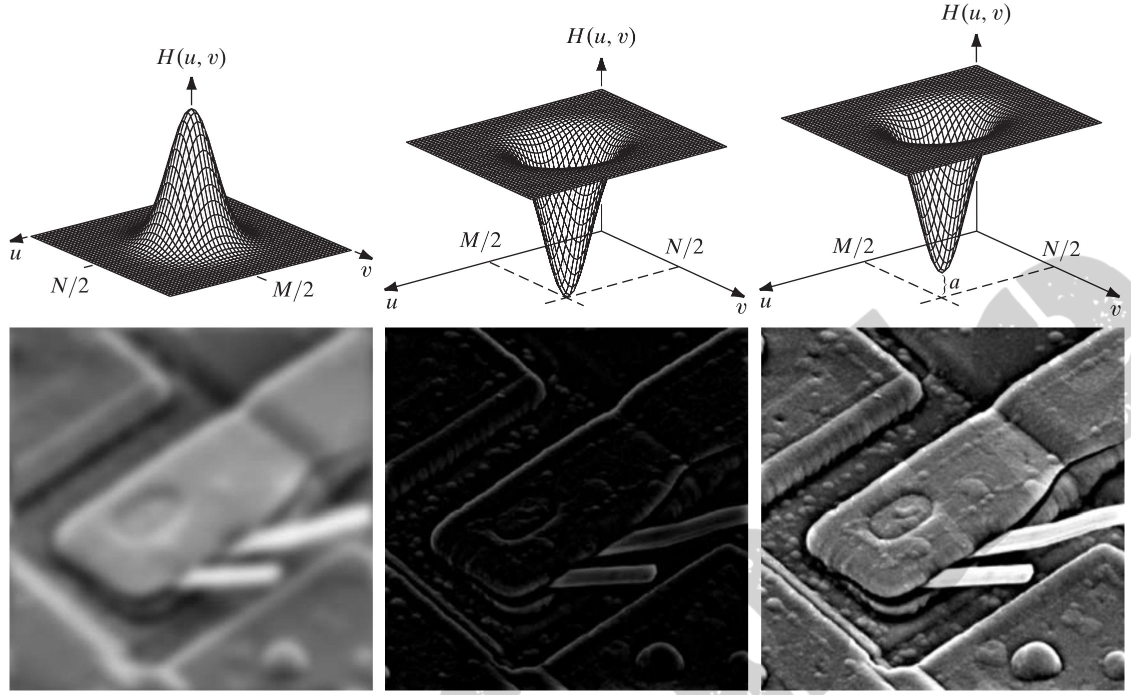


FIGURE 4.31 Top row: frequency domain filters. Bottom row: corresponding filtered images obtained using Eq. (4.7-1). We used $a = 0.85$ in (c) to obtain (f) (the height of the filter itself is 1). Compare (f) with Fig. 4.29(a).

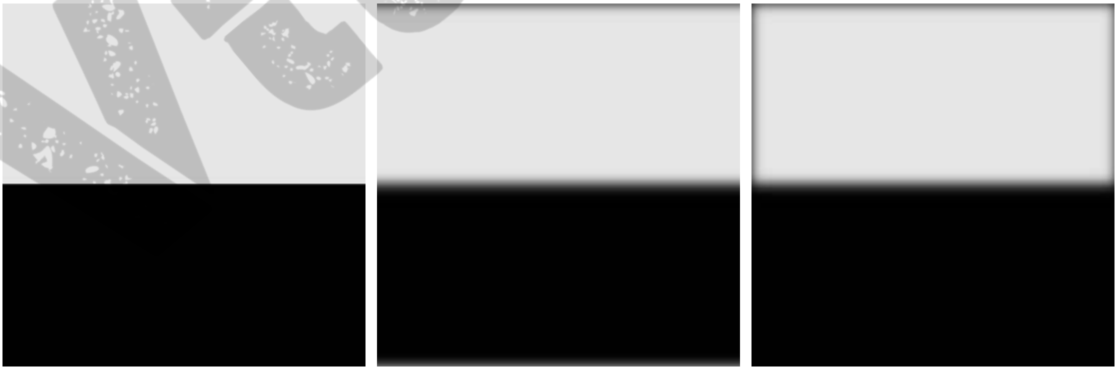
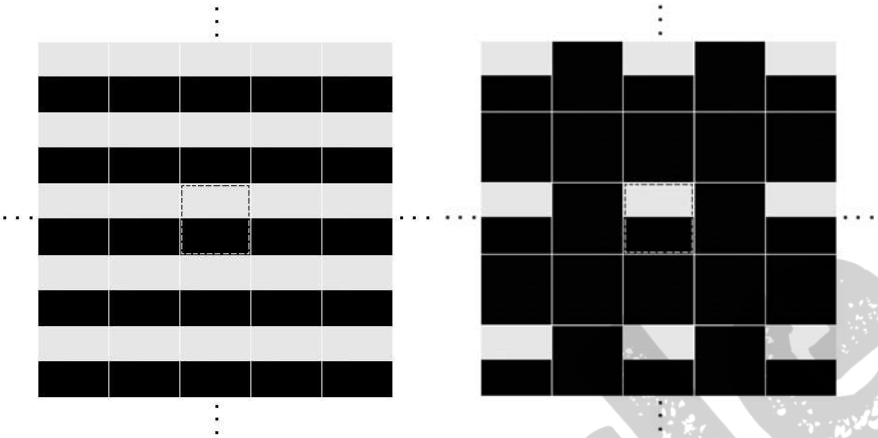


FIGURE 4.32 (a) A simple image. (b) Result of blurring with a Gaussian lowpass filter without padding. (c) Result of lowpass filtering with padding. Compare the light area of the vertical edges in (b) and (c).



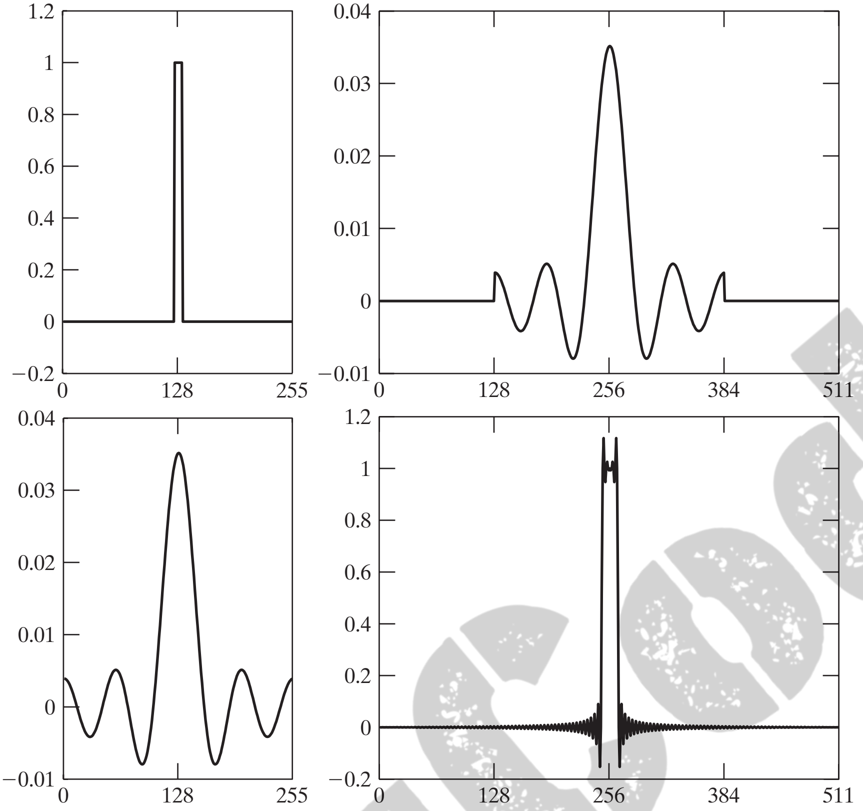
a b

FIGURE 4.33 2-D image periodicity inherent in using the DFT. (a) Periodicity without image padding. (b) Periodicity after padding with 0s (black). The dashed areas in the center correspond to the image in Fig. 4.32(a). (The thin white lines in both images are superimposed for clarity; they are not part of the data.)

dashed image, it will encompass part of the image and also part of the bottom of the periodic image right above it. When a dark and a light region reside under the filter, the result is a mid-gray, blurred output. However, when the filter is passing through the top right side of the image, the filter will encompass only light areas in the image and its right neighbor. The average of a constant is the same constant, so filtering will have no effect in this area, giving the result in Fig. 4.32(b). Padding the image with 0s creates a uniform border around the periodic sequence, as Fig. 4.33(b) shows. Convolving the blurring function with the padded "mosaic" of Fig. 4.33(b) gives the correct result in Fig. 4.32(c). You can see from this example that failure to pad an image can lead to erroneous results. If the purpose of filtering is only for rough visual analysis, the padding step is skipped sometimes.

Thus far, the discussion has centered on padding the input image, but Eq. (4.7-1) also involves a filter that can be specified either in the spatial or in the frequency domain. However, padding is done in the spatial domain, which raises an important question about the relationship between *spatial* padding and filters specified directly in the frequency domain.

At first glance, one could conclude that the way to handle padding of a frequency domain filter is to construct the filter to be of the same size as the image, compute the IDFT of the filter to obtain the corresponding spatial filter, pad that filter in the spatial domain, and then compute its DFT to return to the frequency domain. The 1-D example in Fig. 4.34 illustrates the pitfalls in this approach. Figure 4.34(a) shows a 1-D ideal lowpass filter in the frequency domain. The filter is real and has even symmetry, so we know from property 8 in Table 4.1 that its IDFT will be real and symmetric also. Figure 4.34(b) shows the result of multiplying the elements of the frequency domain filter



a c
b d

FIGURE 4.34 (a) Original filter specified in the (centered) frequency domain. (b) Spatial representation obtained by computing the IDFT of (a). (c) Result of padding (b) to twice its length (note the discontinuities). (d) Corresponding filter in the frequency domain obtained by computing the DFT of (c). Note the ringing caused by the discontinuities in (c). (The curves appear continuous because the points were joined to simplify visual analysis.)

by $(-1)^u$ and computing its IDFT to obtain the corresponding spatial filter. The extremes of this spatial function are not zero so, as Fig. 4.34(c) shows, zero-padding the function created two discontinuities (padding the two ends of the function is the same as padding one end, as long as the total number of zeros used is the same).

To get back to the frequency domain, we compute the DFT of the spatial, padded filter. Figure 4.34(d) shows the result. The discontinuities in the spatial filter created ringing in its frequency domain counterpart, as you would expect from the results in Example 4.1. Viewed another way, we know from that example that the Fourier transform of a box function is a sinc function with frequency components extending to infinity, and we would expect the same behavior from the inverse transform of a box. That is, the spatial representation of an ideal (box) frequency domain filter has components extending to infinity. Therefore, any spatial truncation of the filter to implement zero-padding will introduce discontinuities, which will then in general result in ringing in the frequency domain (truncation can be avoided in this case if it is done at zero crossings, but we are interested in general procedures, and not all filters have zero crossings).

See the end of Section 4.3.3 regarding the definition of an ideal filter.

What the preceding results tell us is that, because we cannot work with an infinite number of components, we cannot use an ideal frequency domain filter [as in

Fig. 4.34(a)] and simultaneously use zero padding to avoid wraparound error. A decision on which limitation to accept is required. Our objective is to work with specified filter shapes in the frequency domain (including ideal filters) without having to be concerned with truncation issues. One approach is to zero-pad images and then create filters in the frequency domain to be of the same size as the padded images (remember, images and filters must be of the same size when using the DFT). Of course, this will result in wraparound error because no padding is used for the filter, but in practice this error is mitigated significantly by the separation provided by the padding of the image, and it is preferable to ringing. Smooth filters (such as those in Fig. 4.31) present even less of a problem. Specifically, then, the approach we will follow in this chapter in order to work with filters of a specified shape directly in the frequency domain is to pad images to size $P \times Q$ and construct filters of the same dimensions. As explained earlier, P and Q are given by Eqs. (4.6-29) and (4.6-30).

We conclude this section by analyzing the phase angle of the filtered transform. Because the DFT is a complex array, we can express it in terms of its real and imaginary parts:

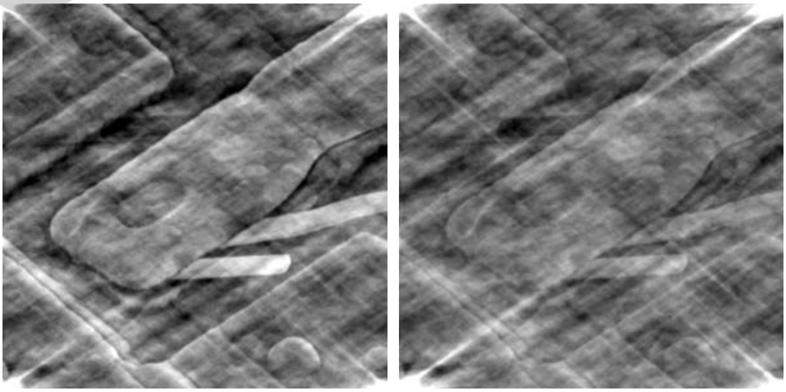
$$F(u, v) = R(u, v) + jI(u, v) \tag{4.7-2}$$

Equation (4.7-1) then becomes

$$g(x, y) = \Im^{-1}\big[H(u, v)R(u, v) + jH(u, v)I(u, v)\big] \tag{4.7-3}$$

The phase angle is not altered by filtering in the manner just described because $H(u, v)$ cancels out when the ratio of the imaginary and real parts is formed in Eq. (4.6-17). Filters that affect the real and imaginary parts equally, and thus have no effect on the phase, are appropriately called *zero-phase-shift* filters. These are the only types of filters considered in this chapter.

Even small changes in the phase angle can have dramatic (usually undesirable) effects on the filtered output. Figure 4.35 illustrates the effect of something as simple as a scalar change. Figure 4.35(a) shows an image resulting from multiplying the angle array in Eq. (4.6-15) by 0.5, without changing



a b

FIGURE 4.35 (a) Image resulting from multiplying by 0.5 the phase angle in Eq. (4.6-15) and then computing the IDFT. (b) The result of multiplying the phase by 0.25. The spectrum was not changed in either of the two cases.

$|F(u, v)|$, and then computing the IDFT. The basic shapes remain unchanged, but the intensity distribution is quite distorted. Figure 4.35(b) shows the result of multiplying the phase by 0.25. The image is almost unrecognizable.

4.7.3 Summary of Steps for Filtering in the Frequency Domain

The material in the previous two sections can be summarized as follows:

1. Given an input image $f(x, y)$ of size $M \times N$, obtain the padding parameters P and Q from Eqs. (4.6-31) and (4.6-32). Typically, we select $P = 2M$ and $Q = 2N$.
2. Form a padded image, $f_p(x, y)$, of size $P \times Q$ by appending the necessary number of zeros to $f(x, y)$.
3. Multiply $f_p(x, y)$ by $(-1)^{x+y}$ to center its transform.
4. Compute the DFT, $F(u, v)$, of the image from step 3.
5. Generate a real, symmetric filter function, $H(u, v)$, of size $P \times Q$ with center at coordinates $(P/2, Q/2)$.[†] Form the product $G(u, v) = H(u, v)F(u, v)$ using array multiplication; that is, $G(i, k) = H(i, k)F(i, k)$.
6. Obtain the processed image:

$$g_p(x, y) = \left\{\text{real}\left[\Im^{-1}[G(u, v)]\right]\right\}(-1)^{x+y}$$

   where the real part is selected in order to ignore parasitic complex components resulting from computational inaccuracies, and the subscript p indicates that we are dealing with padded arrays.
7. Obtain the final processed result, $g(x, y)$, by extracting the $M \times N$ region from the top, left quadrant of $g_p(x, y)$.

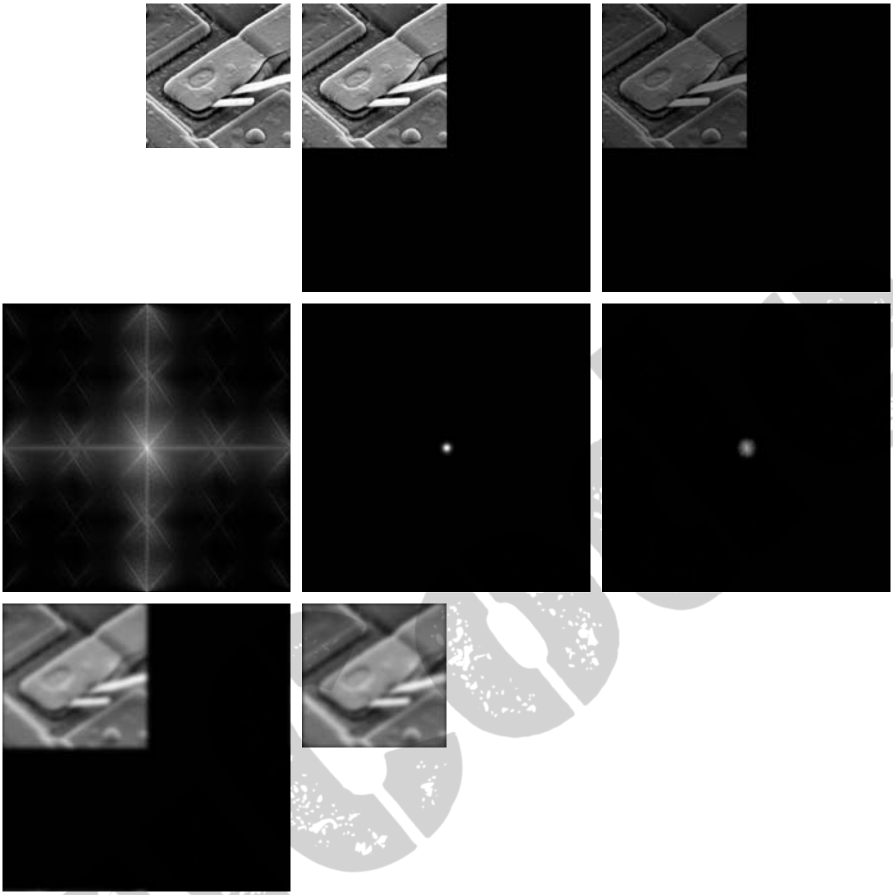
As noted earlier, centering helps in visualizing the filtering process and in generating the filter functions themselves, but centering is not a fundamental requirement.

Figure 4.36 illustrates the preceding steps. The legend in the figure explains the source of each image. If it were enlarged, Fig. 4.36(c) would show black dots interleaved in the image because negative intensities are clipped to 0 for display. Note in Fig. 4.36(h) the characteristic dark border exhibited by lowpass filtered images processed using zero padding.

4.7.4 Correspondence Between Filtering in the Spatial and Frequency Domains

The link between filtering in the spatial and frequency domains is the convolution theorem. In Section 4.7.2, we defined filtering in the frequency domain as the multiplication of a filter function, $H(u, v)$, times $F(u, v)$, the Fourier transform of the input image. Given a filter $H(u, v)$, suppose that we want to find its equivalent representation in the spatial domain. If we let $f(x, y) = \delta(x, y)$, it follows from Table 4.3 that $F(u, v) = 1$. Then, from Eq. (4.7-1), the filtered output is $\Im^{-1}\{H(u, v)\}$. But this is the inverse transform of the frequency domain filter, which is the corresponding filter in the

[†] If $H(u, v)$ is to be generated from a *given* spatial filter, $h(x, y)$, then we form $h_p(x, y)$ by padding the spatial filter to size $P \times Q$, multiply the expanded array by $(-1)^{x+y}$, and compute the DFT of the result to obtain a centered $H(u, v)$. Example 4.15 illustrates this procedure.



a b c
d e f
g h

FIGURE 4.36 (a) An $M \times N$ image, f. (b) Padded image, f_p of size $P \times Q$. (c) Result of multiplying f_p by $(-1)^{x+y}$. (d) Spectrum of F_p. (e) Centered Gaussian lowpass filter, H, of size $P \times Q$. (f) Spectrum of the product HF_p. (g) g_p, the product of $(-1)^{x+y}$ and the real part of the IDFT of HF_p. (h) Final result, g, obtained by cropping the first M rows and N columns of g_p.

spatial domain. Conversely, it follows from a similar analysis and the convolution theorem that, given a spatial filter, we obtain its frequency domain representation by taking the forward Fourier transform of the spatial filter. Therefore, the two filters form a Fourier transform pair:

$$h(x, y) \Leftrightarrow H(u, v) \tag{4.7-4}$$

where $h(x, y)$ is a spatial filter. Because this filter can be obtained from the response of a frequency domain filter to an impulse, $h(x, y)$ sometimes is referred to as the *impulse response* of $H(u, v)$. Also, because all quantities in a discrete implementation of Eq. (4.7-4) are finite, such filters are called *finite impulse response* (FIR) filters. These are the only types of linear spatial filters considered in this book.

We introduced spatial convolution in Section 3.4.1 and discussed its implementation in connection with Eq. (3.4-2), which involved convolving functions of different sizes. When we speak of spatial convolution in terms of the convolution theorem and the DFT, it is implied that we are convolving periodic functions, as explained in Fig. 4.28. For this reason, as explained earlier, Eq. (4.6-23) is referred to as *circular convolution*. Furthermore, convolution in the context of the DFT involves functions of the same size, whereas in Eq. (3.4-2) the functions typically are of different sizes.

In practice, we prefer to implement convolution filtering using Eq. (3.4-2) with small filter masks because of speed and ease of implementation in hardware and/or firmware. However, filtering concepts are more intuitive in the frequency domain. One way to take advantage of the properties of both domains is to specify a filter in the frequency domain, compute its IDFT, and then use the resulting, full-size spatial filter as a *guide* for constructing smaller spatial filter masks (more formal approaches are mentioned in Section 4.11.4). This is illustrated next. Later in this section, we illustrate also the converse, in which a small spatial filter is given and we obtain its full-size frequency domain representation. This approach is useful for analyzing the behavior of small spatial filters in the frequency domain. Keep in mind during the following discussion that the Fourier transform and its inverse are linear processes (Problem 4.14), so the discussion is limited to linear filtering.

In the following discussion, we use Gaussian filters to illustrate how frequency domain filters can be used as guides for specifying the coefficients of some of the small masks discussed in Chapter 3. Filters based on Gaussian functions are of particular interest because, as noted in Table 4.3, both the forward and inverse Fourier transforms of a Gaussian function are real Gaussian functions. We limit the discussion to 1-D to illustrate the underlying principles. Two-dimensional Gaussian filters are discussed later in this chapter.

Let $H(u)$ denote the 1-D frequency domain Gaussian filter:

$$H(u) = Ae^{-u^2/2\sigma^2} \tag{4.7-5}$$

where σ is the standard deviation of the Gaussian curve. The corresponding filter in the spatial domain is obtained by taking the inverse Fourier transform of $H(u)$ (Problem 4.31):

$$h(x) = \sqrt{2\pi}\sigma Ae^{-2\pi^2\sigma^2x^2} \tag{4.7-6}$$

These equations[†] are important for two reasons: (1) They are a Fourier transform pair, both components of which are Gaussian and *real*. This facilitates analysis because we do not have to be concerned with complex numbers. In addition, Gaussian curves are intuitive and easy to manipulate. (2) The functions behave reciprocally. When $H(u)$ has a broad profile (large value of σ),

[†] As mentioned in Table 4.3, closed forms for the forward and inverse Fourier transforms of Gaussians are valid only for continuous functions. To use discrete formulations we simply sample the continuous Gaussian transforms. Our use of discrete variables here implies that we are dealing with sampled transforms.

$h(x)$ has a narrow profile, and vice versa. In fact, as σ approaches infinity, $H(u)$ tends toward a constant function and $h(x)$ tends toward an impulse, which implies no filtering in the frequency and spatial domains, respectively.

Figures 4.37(a) and (b) show plots of a Gaussian lowpass filter in the frequency domain and the corresponding lowpass filter in the spatial domain. Suppose that we want to use the shape of $h(x)$ in Fig. 4.37(b) as a *guide* for specifying the coefficients of a small spatial mask. The key similarity between the two filters is that all their values are positive. Thus, we conclude that we can implement lowpass filtering in the spatial domain by using a mask with all positive coefficients (as we did in Section 3.5.1). For reference, Fig. 4.37(b) shows two of the masks discussed in that section. Note the reciprocal relationship between the width of the filters, as discussed in the previous paragraph. The narrower the frequency domain filter, the more it will attenuate the low frequencies, resulting in increased blurring. In the spatial domain, this means that a larger mask must be used to increase blurring, as illustrated in Example 3.13.

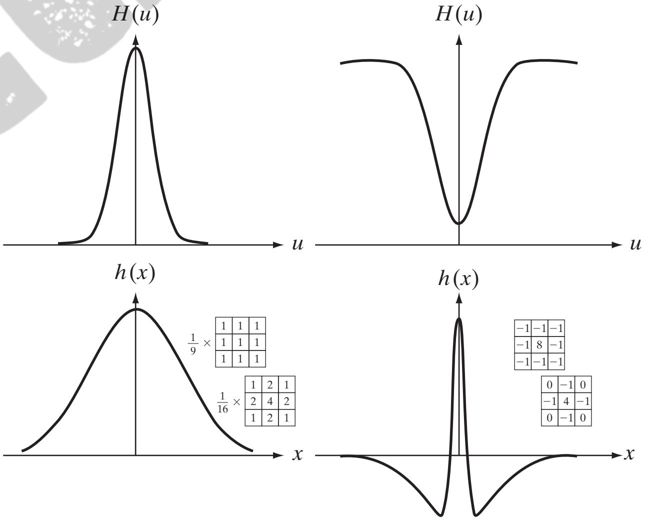
More complex filters can be constructed using the basic Gaussian function of Eq. (4.7-5). For example, we can construct a highpass filter as the *difference* of Gaussians:

$$H(u) = Ae^{-u^2/2\sigma_1^2} - Be^{-u^2/2\sigma_2^2} \tag{4.7-7}$$

with $A \geq B$ and $\sigma_1 > \sigma_2$. The corresponding filter in the spatial domain is

$$h(x) = \sqrt{2\pi}\sigma_1 Ae^{-2\pi^2\sigma_1^2x^2} - \sqrt{2\pi}\sigma_2 Be^{-2\pi^2\sigma_2^2x^2} \tag{4.7-8}$$

Figures 4.37(c) and (d) show plots of these two equations. We note again the reciprocity in width, but the most important feature here is that $h(x)$ has a positive center term with negative terms on either side. The small masks shown in



a c
b d

FIGURE 4.37 (a) A 1-D Gaussian lowpass filter in the frequency domain. (b) Spatial lowpass filter corresponding to (a). (c) Gaussian highpass filter in the frequency domain. (d) Spatial highpass filter corresponding to (c). The small 2-D masks shown are spatial filters we used in Chapter 3.

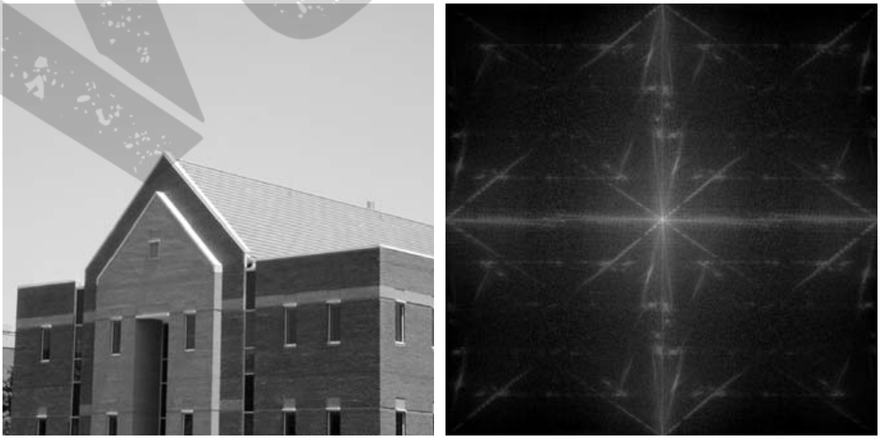
Fig. 4.37(d) "capture" this property. These two masks were used in Chapter 3 as sharpening filters, which we now know are highpass filters.

Although we have gone through significant effort to get here, be assured that it is impossible to truly understand filtering in the frequency domain without the foundation we have just established. In practice, the frequency domain can be viewed as a "laboratory" in which we take advantage of the correspondence between frequency content and image appearance. As is demonstrated numerous times later in this chapter, some tasks that would be exceptionally difficult or impossible to formulate directly in the spatial domain become almost trivial in the frequency domain. Once we have selected a specific filter via experimentation in the frequency domain, the actual implementation of the method usually is done in the spatial domain. One approach is to specify small spatial masks that attempt to capture the "essence" of the full filter function in the spatial domain, as we explained in Fig. 4.37. A more formal approach is to design a 2-D digital filter by using approximations based on mathematical or statistical criteria. We touch on this point again in Section 4.11.4.

EXAMPLE 4.15: Obtaining a frequency domain filter from a small spatial mask.

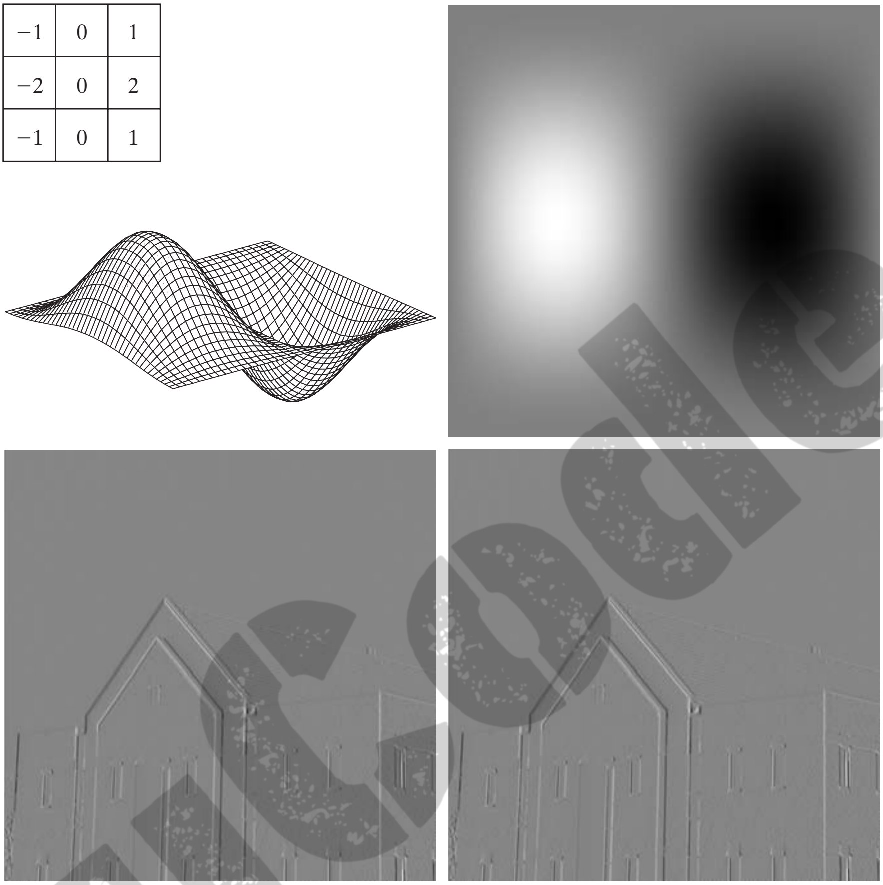
■ In this example, we start with a spatial mask and show how to generate its corresponding filter in the frequency domain. Then, we compare the filtering results obtained using frequency domain and spatial techniques. This type of analysis is useful when one wishes to compare the performance of given spatial masks against one or more "full" filter candidates in the frequency domain, or to gain deeper understanding about the performance of a mask. To keep matters simple, we use the 3×3 Sobel vertical edge detector from Fig. 3.41(e). Figure 4.38(a) shows a 600×600 pixel image, $f(x, y)$, that we wish to filter, and Fig. 4.38(b) shows its spectrum.

Figure 4.39(a) shows the Sobel mask, $h(x, y)$ (the perspective plot is explained below). Because the input image is of size 600×600 pixels and the filter is of size 3×3 we avoid wraparound error by padding f and h to size



a b

FIGURE 4.38 (a) Image of a building, and (b) its spectrum.



−1	0	1
−2	0	2
−1	0	1

a b
c d

FIGURE 4.39 (a) A spatial mask and perspective plot of its corresponding frequency domain filter. (b) Filter shown as an image. (c) Result of filtering Fig. 4.38(a) in the frequency domain with the filter in (b). (d) Result of filtering the same image with the spatial filter in (a). The results are identical.

602×602 pixels, according to Eqs. (4.6-29) and (4.6-30). The Sobel mask exhibits odd symmetry, provided that it is embedded in an array of zeros of even size (see Example 4.10). To maintain this symmetry, we place $h(x, y)$ so that its center is at the center of the 602×602 padded array. This is an important aspect of filter generation. If we preserve the odd symmetry with respect to the padded array in forming $h_p(x, y)$, we know from property 9 in Table 4.1 that $H(u, v)$ will be purely imaginary. As we show at the end of this example, this will yield results that are identical to filtering the image spatially using $h(x, y)$. If the symmetry were not preserved, the results would no longer be same.

The procedure used to generate $H(u, v)$ is: (1) multiply $h_p(x, y)$ by $(-1)^{x+y}$ to center the frequency domain filter; (2) compute the forward DFT of the result in (1); (3) set the real part of the resulting DFT to 0 to account for parasitic real parts (we know that $H(u, v)$ has to be purely imaginary); and (4) multiply the result by $(-1)^{u+v}$. This last step reverses the multiplication of $H(u, v)$ by $(-1)^{u+v}$, which is implicit when $h(x, y)$ was moved to the center of $h_p(x, y)$. Figure 4.39(a) shows a perspective plot of $H(u, v)$, and Fig. 4.39(b) shows

$H(u, v)$ as an image. As, expected, the function is odd, thus the antisymmetry about its center. Function $H(u, v)$ is used as any other frequency domain filter in the procedure outlined in Section 4.7.3.

Figure 4.39(c) is the result of using the filter just obtained in the procedure outlined in Section 4.7.3 to filter the image in Fig. 4.38(a). As expected from a derivative filter, edges are enhanced and all the constant intensity areas are reduced to zero (the grayish tone is due to scaling for display). Figure 4.39(d) shows the result of filtering the same image in the spatial domain directly, using $h(x, y)$ in the procedure outlined in Section 3.6.4. The results are identical. ■

4.8 Image Smoothing Using Frequency Domain Filters

The remainder of this chapter deals with various filtering techniques in the frequency domain. We begin with lowpass filters. Edges and other sharp intensity transitions (such as noise) in an image contribute significantly to the high-frequency content of its Fourier transform. Hence, smoothing (blurring) is achieved in the frequency domain by high-frequency attenuation; that is, by *lowpass* filtering. In this section, we consider three types of lowpass filters: ideal, Butterworth, and Gaussian. These three categories cover the range from very sharp (ideal) to very smooth (Gaussian) filtering. The Butterworth filter has a parameter called the *filter order*. For high order values, the Butterworth filter approaches the ideal filter. For lower order values, the Butterworth filter is more like a Gaussian filter. Thus, the Butterworth filter may be viewed as providing a transition between two "extremes." All filtering in this section follows the procedure outlined in Section 4.7.3, so all filter functions, $H(u, v)$, are understood to be discrete functions of size $P \times Q$; that is, the discrete frequency variables are in the range $u = 0, 1, 2, \dots, P - 1$ and $v = 0, 1, 2, \dots, Q - 1$.

4.8.1 Ideal Lowpass Filters

A 2-D lowpass filter that passes without attenuation all frequencies within a circle of radius D_0 from the origin and "cuts off" all frequencies outside this circle is called an *ideal lowpass filter* (ILPF); it is specified by the function

$$H(u, v) = \begin{cases} 1 & \text{if } D(u, v) \le D_0 \\ 0 & \text{if } D(u, v) > D_0 \end{cases} \tag{4.8-1}$$

where D_0 is a positive constant and $D(u, v)$ is the distance between a point (u, v) in the frequency domain and the center of the frequency rectangle; that is,

$$D(u, v) = \left[(u - P/2)^2 + (v - Q/2)^2\right]^{1/2} \tag{4.8-2}$$

where, as before, P and Q are the padded sizes from Eqs. (4.6-31) and (4.6-32). Figure 4.40(a) shows a perspective plot of $H(u, v)$ and Fig. 4.40(b) shows the filter displayed as an image. As mentioned in Section 4.3.3, the name *ideal* indicates that all frequencies on or inside a circle of radius D_0 are passed

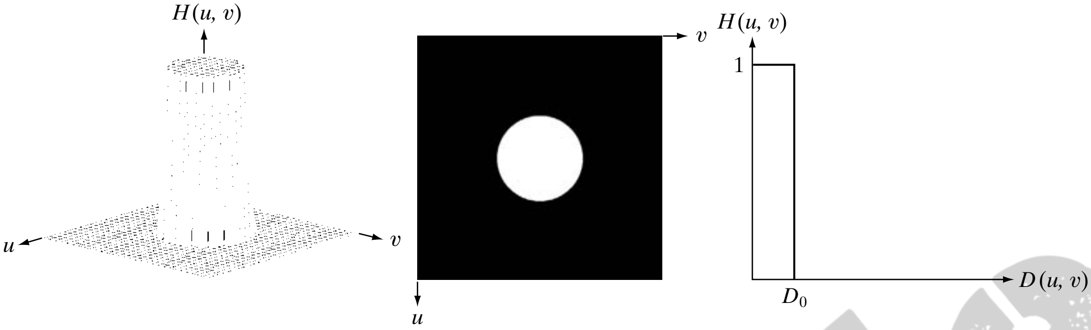


FIGURE 4.40 (a) Perspective plot of an ideal lowpass-filter transfer function. (b) Filter displayed as an image. (c) Filter radial cross section.

without attenuation, whereas all frequencies outside the circle are completely attenuated (filtered out). The ideal lowpass filter is radially symmetric about the origin, which means that the filter is completely defined by a radial cross section, as Fig. 4.40(c) shows. Rotating the cross section by 360° yields the filter in 2-D.

For an ILPF cross section, the point of transition between $H(u, v) = 1$ and $H(u, v) = 0$ is called the *cutoff frequency*. In the case of Fig. 4.40, for example, the cutoff frequency is D_0. The sharp cutoff frequencies of an ILPF cannot be realized with electronic components, although they certainly can be simulated in a computer. The effects of using these "nonphysical" filters on a digital image are discussed later in this section.

The lowpass filters introduced in this chapter are compared by studying their behavior as a function of the same cutoff frequencies. One way to establish a set of standard cutoff frequency loci is to compute circles that enclose specified amounts of total image power P_T. This quantity is obtained by summing the components of the power spectrum of the padded images at each point (u, v), for $u = 0, 1, \ldots, P - 1$ and $v = 0, 1, \ldots, Q - 1$; that is,

$$P_T = \sum_{u=0}^{P-1} \sum_{v=0}^{Q-1} P(u, v) \tag{4.8-3}$$

where $P(u, v)$ is given in Eq. (4.6-18). If the DFT has been centered, a circle of radius D_0 with origin at the center of the frequency rectangle encloses α percent of the power, where

$$\alpha = 100\left[\sum_{u} \sum_{v} P(u, v)/P_T\right] \tag{4.8-4}$$

and the summation is taken over values of (u, v) that lie inside the circle or on its boundary.

Figures 4.41(a) and (b) show a test pattern image and its spectrum. The circles superimposed on the spectrum have radii of 10, 30, 60, 160, and 460 pixels, respectively. These circles enclose α percent of the image power, for $\alpha = 87.0$, 93.1, 95.7, 97.8, and 99.2%, respectively. The spectrum falls off rapidly, with 87% of the total power being enclosed by a relatively small circle of radius 10.

EXAMPLE 4.16: Image smoothing using an ILPF.

■ Figure 4.42 shows the results of applying ILPFs with cutoff frequencies at the radii shown in Fig. 4.41(b). Figure 4.42(b) is useless for all practical purposes, unless the objective of blurring is to eliminate all detail in the image, except the "blobs" representing the largest objects. The severe blurring in this image is a clear indication that most of the sharp detail information in the picture is contained in the 13% power removed by the filter. As the filter radius increases, less and less power is removed, resulting in less blurring. Note that the images in Figs. 4.42(c) through (e) are characterized by "ringing," which becomes finer in texture as the amount of high frequency content removed decreases. Ringing is visible even in the image [Fig. 4.42(e)] in which only 2% of the total power was removed. This ringing behavior is a characteristic of ideal filters, as you will see shortly. Finally, the result for $\alpha = 99.2$ shows very slight blurring in the noisy squares but, for the most part, this image is quite close to the original. This indicates that little edge information is contained in the upper 0.8% of the spectrum power in this particular case.

It is clear from this example that ideal lowpass filtering is not very practical. However, it is useful to study their behavior as part of our development of

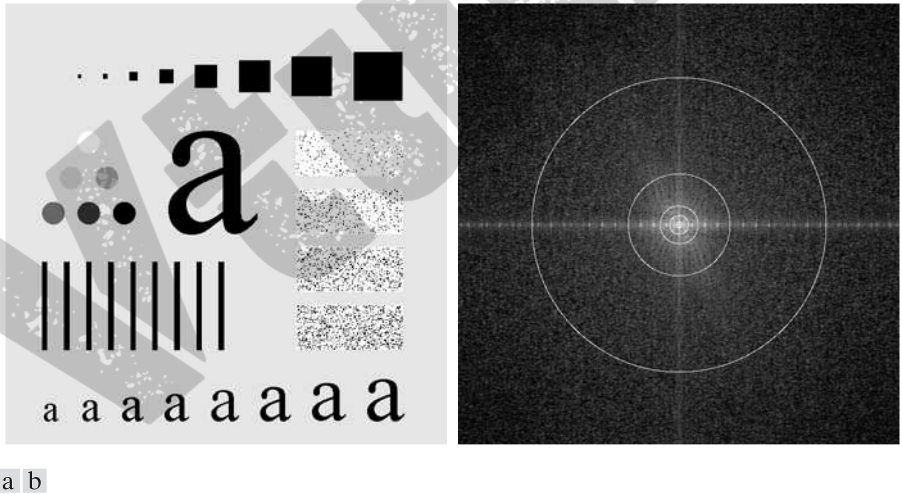
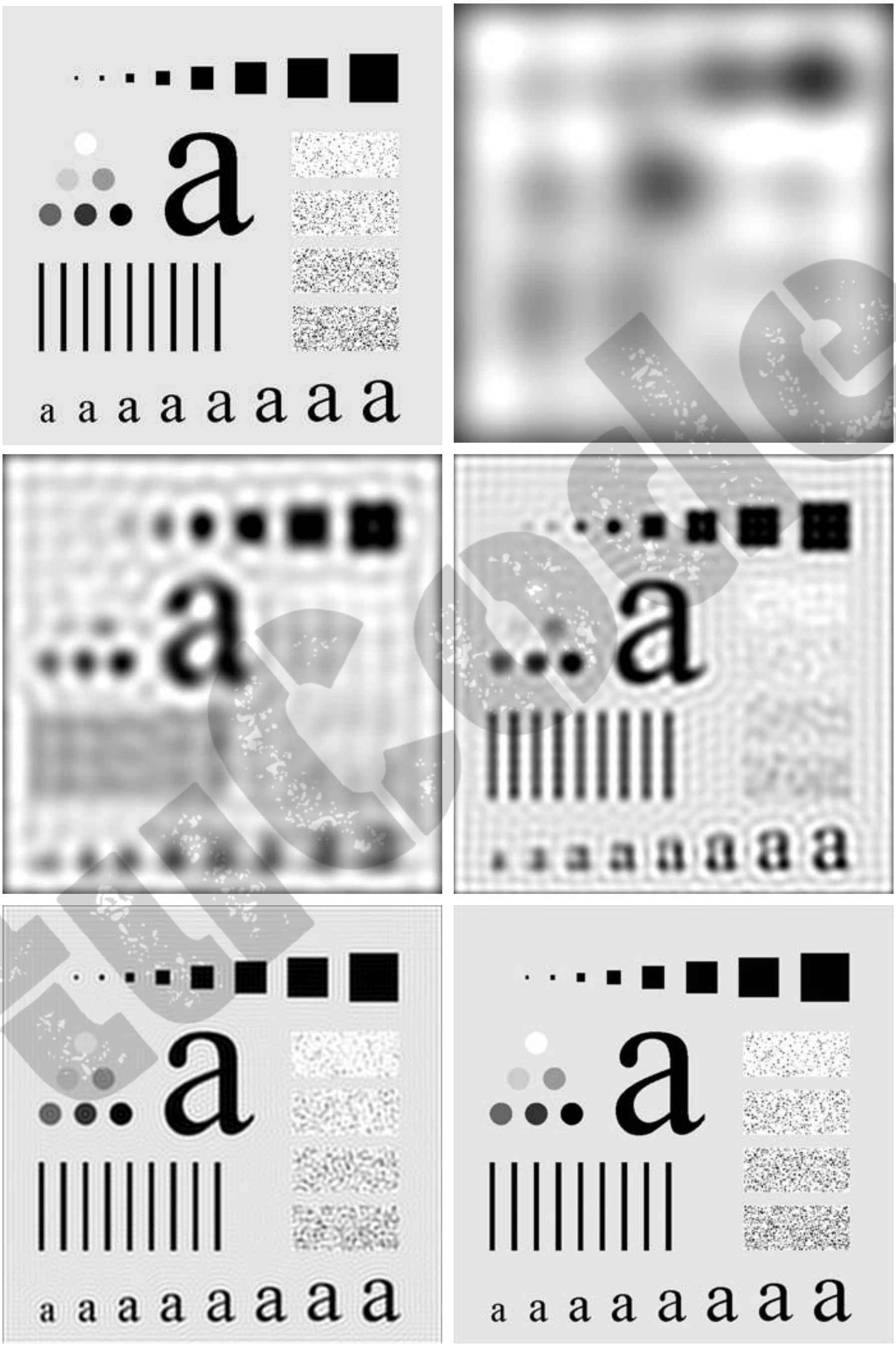


FIGURE 4.41 (a) Test pattern of size 688 × 688 pixels, and (b) its Fourier spectrum. The spectrum is double the image size due to padding but is shown in half size so that it fits in the page. The superimposed circles have radii equal to 10, 30, 60, 160, and 460 with respect to the full-size spectrum image. These radii enclose 87.0, 93.1, 95.7, 97.8, and 99.2% of the padded image power, respectively.



a b
c d
e f

FIGURE 4.42 (a) Original image. (b)–(f) Results of filtering using ILPFs with cutoff frequencies set at radii values 10, 30, 60, 160, and 460, as shown in Fig. 4.41(b). The power removed by these filters was 13, 6.9, 4.3, 2.2, and 0.8% of the total, respectively.

filtering concepts. Also, as shown in the discussion that follows, some interesting insight is gained by attempting to explain the ringing property of ILPFs in the spatial domain. ■

The blurring and ringing properties of ILPFs can be explained using the convolution theorem. Figure 4.43(a) shows the spatial representation, $h(x, y)$, of an ILPF of radius 10, and Fig. 4.43(b) shows the intensity profile of a line passing through the center of the image. Because a cross section of the ILPF in the frequency domain looks like a box filter, it is not unexpected that a cross section of the corresponding spatial filter has the shape of a sinc function. Filtering in the spatial domain is done by convolving $h(x, y)$ with the image. Imagine each pixel in the image being a discrete impulse whose strength is proportional to the intensity of the image at that location. Convolving a sinc with an impulse copies the sinc at the location of the impulse. The center lobe of the sinc is the principal cause of blurring, while the outer, smaller lobes are mainly responsible for ringing. Convolving the sinc with every pixel in the image provides a nice model for explaining the behavior of ILPFs. Because the "spread" of the sinc function is inversely proportional to the radius of $H(u, v)$, the larger D_0 becomes, the more the spatial sinc approaches an impulse which, in the limit, causes no blurring at all when convolved with the image. This type of reciprocal behavior should be routine to you by now. In the next two sections, we show that it is possible to achieve blurring with little or no ringing, which is an important objective in lowpass filtering.

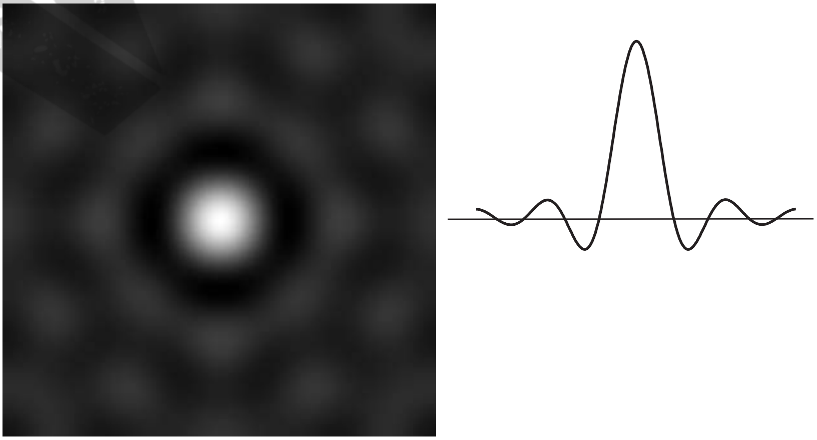
4.8.2 Butterworth Lowpass Filters

The transfer function of a Butterworth lowpass filter (BLPF) of order n, and with cutoff frequency at a distance D_0 from the origin, is defined as

$$H(u, v) = \frac{1}{1 + [D(u, v)/D_0]^{2n}} \tag{4.8-5}$$

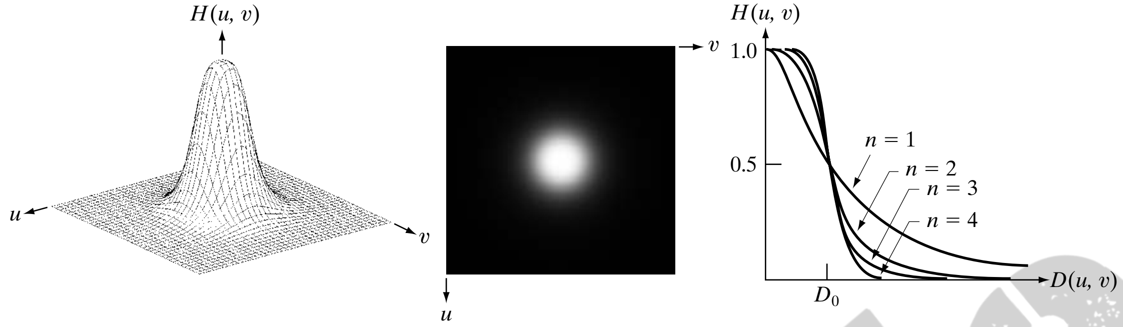
where $D(u, v)$ is given by Eq. (4.8-2). Figure 4.44 shows a perspective plot, image display, and radial cross sections of the BLPF function.

The transfer function of the Butterworth lowpass filter normally is written as the square root of our expression. However, our interest here is in the basic *form* of the filter, so we exclude the square root for computational convenience.



a b

FIGURE 4.43 (a) Representation in the spatial domain of an ILPF of radius 5 and size 1000 × 1000. (b) Intensity profile of a horizontal line passing through the center of the image.



a b c

FIGURE 4.44 (a) Perspective plot of a Butterworth lowpass-filter transfer function. (b) Filter displayed as an image. (c) Filter radial cross sections of orders 1 through 4.

Unlike the ILPF, the BLPF transfer function does not have a sharp discontinuity that gives a clear cutoff between passed and filtered frequencies. For filters with smooth transfer functions, defining a cutoff frequency locus at points for which $H(u, v)$ is down to a certain fraction of its maximum value is customary. In Eq. (4.8-5), (down 50% from its maximum value of 1) when $D(u, v) = D_0$.

EXAMPLE 4.17: Image smoothing with a Butterworth lowpass filter.

■ Figure 4.45 shows the results of applying the BLPF of Eq. (4.8-5) to Fig. 4.45(a), with $n = 2$ and D_0 equal to the five radii in Fig. 4.41(b). Unlike the results in Fig. 4.42 for the ILPF, we note here a smooth transition in blurring as a function of increasing cutoff frequency. Moreover, no ringing is visible in any of the images processed with this particular BLPF, a fact attributed to the filter's smooth transition between low and high frequencies. ■

A BLPF of order 1 has no ringing in the spatial domain. Ringing generally is imperceptible in filters of order 2, but can become significant in filters of higher order. Figure 4.46 shows a comparison between the *spatial* representation of BLPFs of various orders (using a cutoff frequency of 5 in all cases). Shown also is the intensity profile along a horizontal scan line through the center of each filter. These filters were obtained and displayed using the same procedure used to generate Fig. 4.43. To facilitate comparisons, additional enhancing with a gamma transformation [see Eq. (3.2-3)] was applied to the images of Fig. 4.46. The BLPF of order 1 [Fig. 4.46(a)] has neither ringing nor negative values. The filter of order 2 does show mild ringing and small negative values, but they certainly are less pronounced than in the ILPF. As the remaining images show, ringing in the BLPF becomes significant for higher-order filters. A Butterworth filter of order 20 exhibits characteristics similar to those of the ILPF (in the limit, both filters are identical). BLPFs of order 2 are a good compromise between effective lowpass filtering and acceptable ringing.

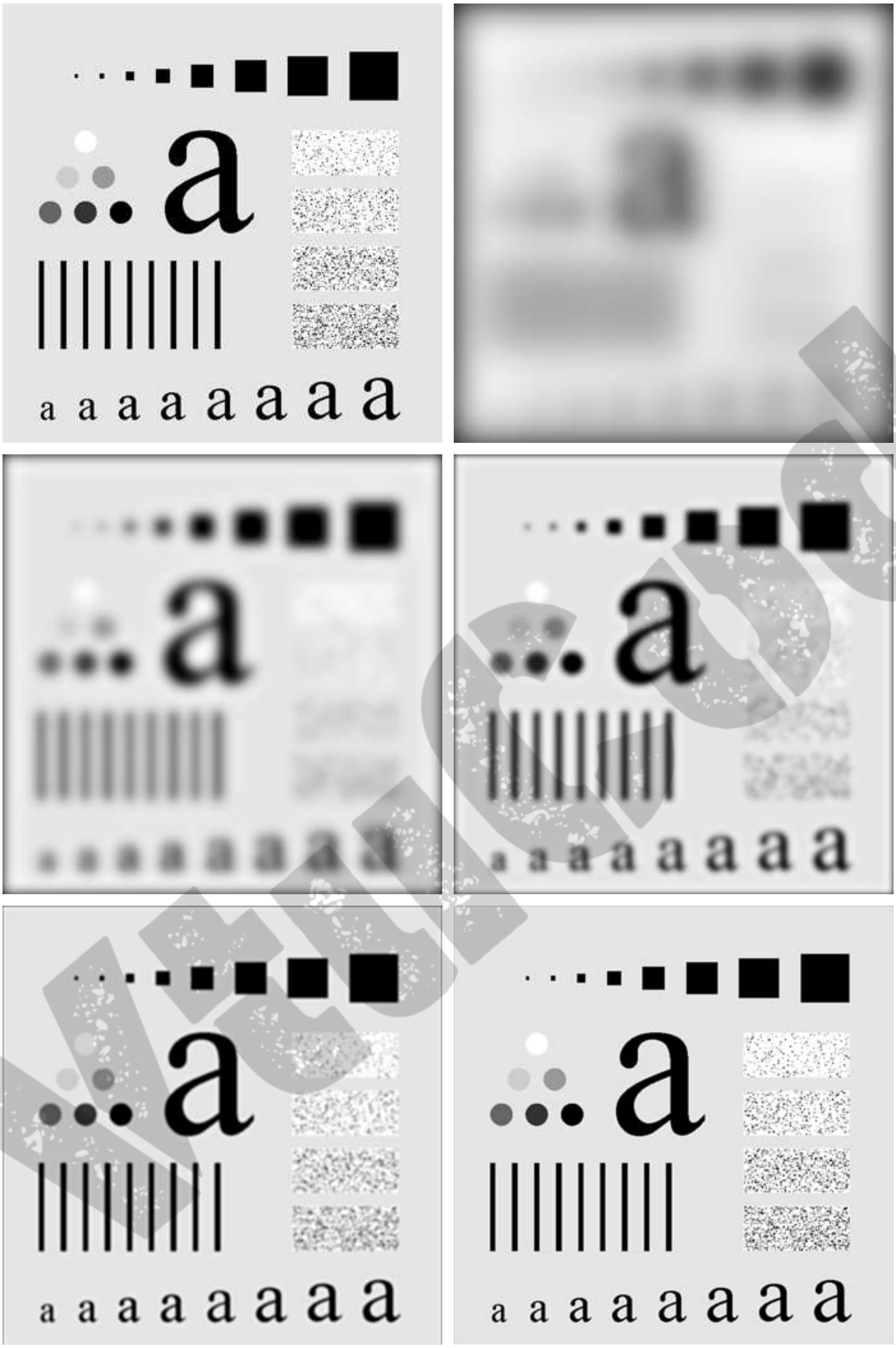


FIGURE 4.45 (a) Original image. (b)–(f) Results of filtering using BLPFs of order 2, with cutoff frequencies at the radii shown in Fig. 4.41. Compare with Fig. 4.42.

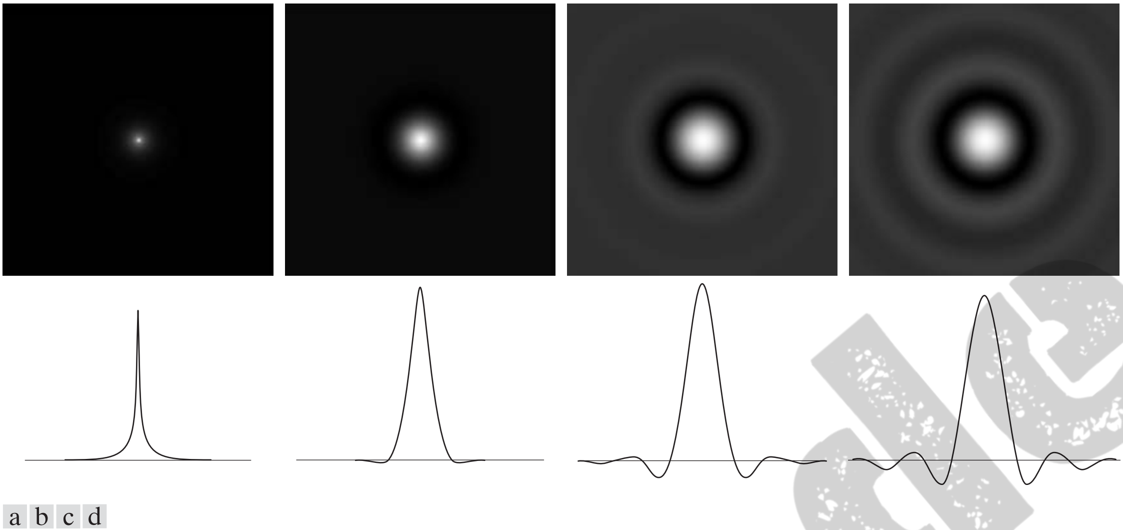


FIGURE 4.46 (a)–(d) Spatial representation of BLPFs of order 1, 2, 5, and 20, and corresponding intensity profiles through the center of the filters (the size in all cases is 1000 × 1000 and the cutoff frequency is 5). Observe how ringing increases as a function of filter order.

4.8.3 Gaussian Lowpass Filters

Gaussian lowpass filters (GLPFs) of one dimension were introduced in Section 4.7.4 as an aid in exploring some important relationships between the spatial and frequency domains. The form of these filters in two dimensions is given by

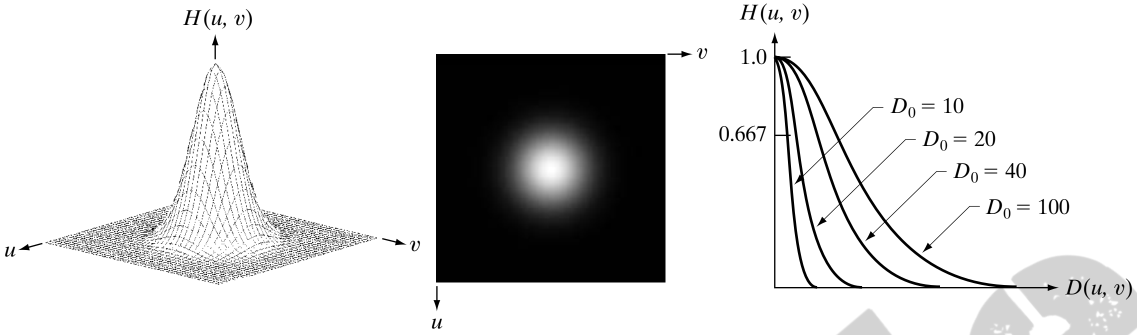
$$H(u, v) = e^{-D^2(u,v)/2\sigma^2} \tag{4.8-6}$$

where, as in Eq. (4.8-2), $D(u, v)$ is the distance from the center of the frequency rectangle. Here we do not use a multiplying constant as in Section 4.7.4 in order to be consistent with the filters discussed in the present section, whose highest value is 1. As before, σ is a measure of spread about the center. By letting $\sigma = D_0$, we can express the filter using the notation of the other filters in this section:

$$H(u, v) = e^{-D^2(u,v)/2D_0^2} \tag{4.8-7}$$

where D_0 is the cutoff frequency. When $D(u, v) = D_0$, the GLPF is down to 0.607 of its maximum value.

As Table 4.3 shows, the inverse Fourier transform of the GLPF is Gaussian also. This means that a spatial Gaussian filter, obtained by computing the IDFT of Eq. (4.8-6) or (4.8-7), will have no ringing. Figure 4.47 shows a perspective plot, image display, and radial cross sections of a GLPF function, and Table 4.4 summarizes the lowpass filters discussed in this section.



a b c

FIGURE 4.47 (a) Perspective plot of a GLPF transfer function. (b) Filter displayed as an image. (c) Filter radial cross sections for various values of D_0.

TABLE 4.4
Lowpass filters. D_0 is the cutoff frequency and n is the order of the Butterworth filter.

Ideal	Butterworth	Gaussian
$H(u, v) = \begin{cases} 1 & \text{if } D(u, v) \leq D_0 \\ 0 & \text{if } D(u, v) > D_0 \end{cases}$	$H(u, v) = \dfrac{1}{1 + [D(u, v)/D_0]^{2n}}$	$H(u, v) = e^{-D^2(u,v)/2D_0^2}$

EXAMPLE 4.18: Image smoothing with a Gaussian lowpass filter.

■ Figure 4.48 shows the results of applying the GLPF of Eq. (4.8-7) to Fig. 4.48(a), with D_0 equal to the five radii in Fig. 4.41(b). As in the case of the BLPF of order 2 (Fig. 4.45), we note a smooth transition in blurring as a function of increasing cutoff frequency. The GLPF achieved slightly less smoothing than the BLPF of order 2 for the same value of cutoff frequency, as can be seen, for example, by comparing Figs. 4.45(c) and 4.48(c). This is expected, because the profile of the GLPF is not as "tight" as the profile of the BLPF of order 2. However, the results are quite comparable, and we are assured of no ringing in the case of the GLPF. This is an important characteristic in practice, especially in situations (e.g., medical imaging) in which any type of artifact is unacceptable. In cases where tight control of the transition between low and high frequencies about the cutoff frequency are needed, then the BLPF presents a more suitable choice. The price of this additional control over the filter profile is the possibility of ringing. ■

4.8.4 Additional Examples of Lowpass Filtering

In the following discussion, we show several practical applications of lowpass filtering in the frequency domain. The first example is from the field of machine perception with application to character recognition; the second is from the printing and publishing industry; and the third is related to processing

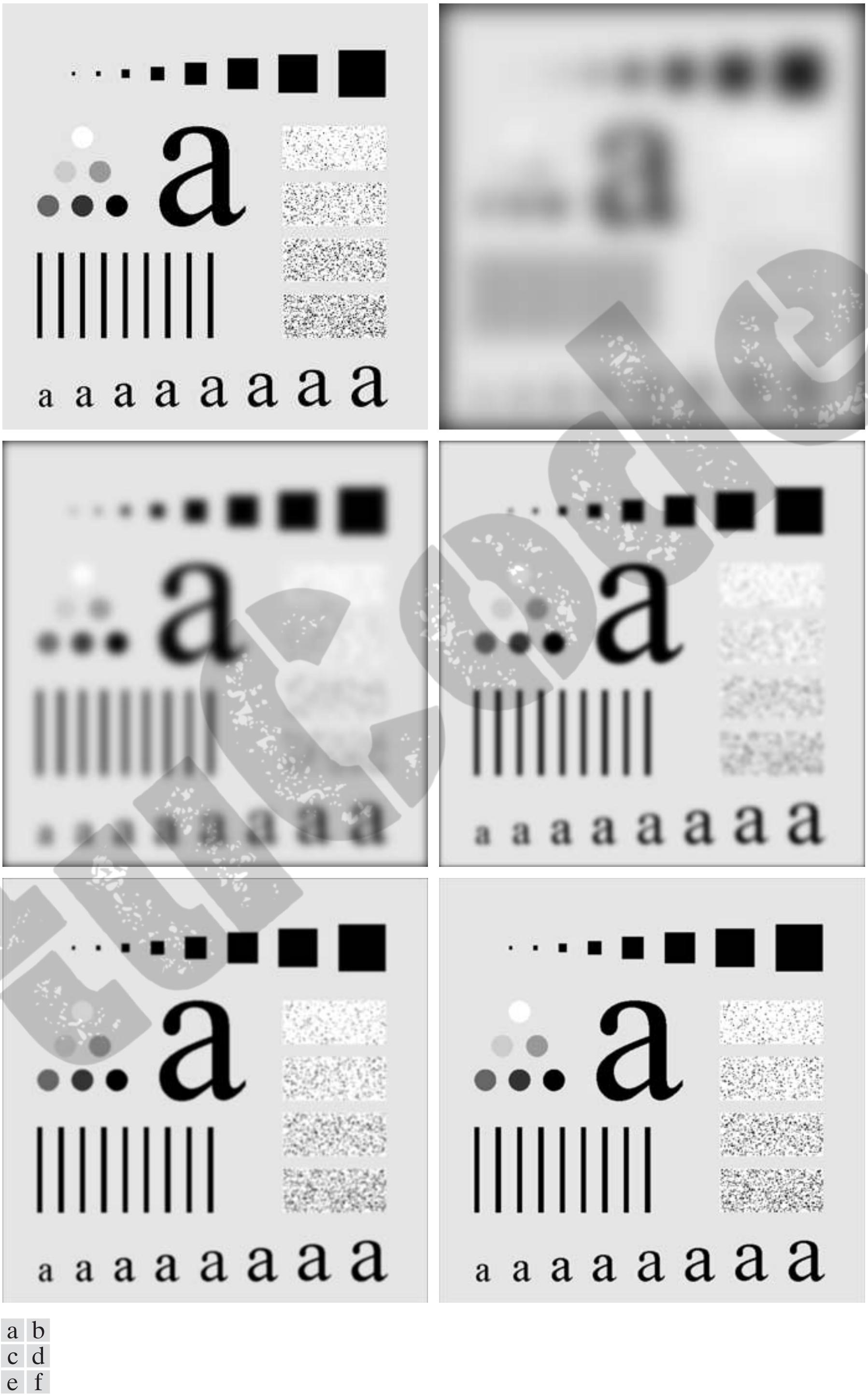
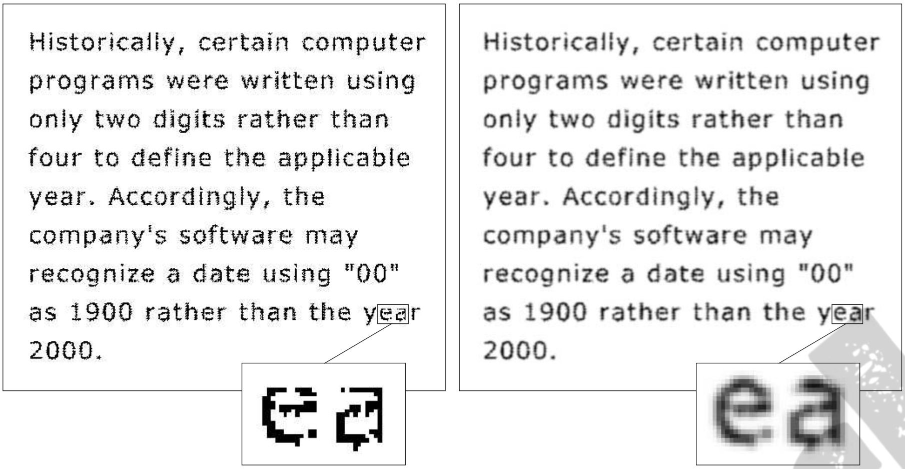


FIGURE 4.48 (a) Original image. (b)–(f) Results of filtering using GLPFs with cutoff frequencies at the radii shown in Fig. 4.41. Compare with Figs. 4.42 and 4.45.



Historically, certain computer programs were written using only two digits rather than four to define the applicable year. Accordingly, the company's software may recognize a date using "00" as 1900 rather than the year 2000.

Historically, certain computer programs were written using only two digits rather than four to define the applicable year. Accordingly, the company's software may recognize a date using "00" as 1900 rather than the year 2000.

a b

FIGURE 4.49 (a) Sample text of low resolution (note broken characters in magnified view). (b) Result of filtering with a GLPF (broken character segments were joined).

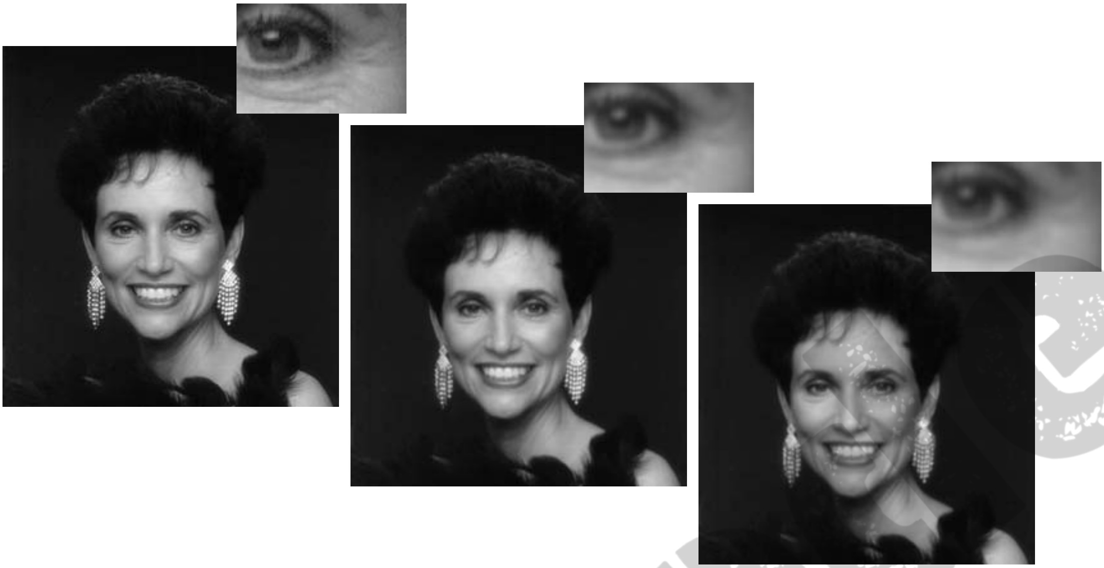
satellite and aerial images. Similar results can be obtained using the lowpass spatial filtering techniques discussed in Section 3.5.

Figure 4.49 shows a sample of text of poor resolution. One encounters text like this, for example, in fax transmissions, duplicated material, and historical records. This particular sample is free of additional difficulties like smudges, creases, and torn sections. The magnified section in Fig. 4.49(a) shows that the characters in this document have distorted shapes due to lack of resolution, and many of the characters are broken. Although humans fill these gaps visually without difficulty, machine recognition systems have real difficulties reading broken characters. One approach for handling this problem is to bridge small gaps in the input image by blurring it. Figure 4.49(b) shows how well characters can be “repaired” by this simple process using a Gaussian lowpass filter with $D_0 = 80$. The images are of size 444×508 pixels.

Lowpass filtering is a staple in the printing and publishing industry, where it is used for numerous preprocessing functions, including unsharp masking, as discussed in Section 3.6.3. “Cosmetic” processing is another use of lowpass filtering prior to printing. Figure 4.50 shows an application of lowpass filtering for producing a smoother, softer-looking result from a sharp original. For human faces, the typical objective is to reduce the sharpness of fine skin lines and small blemishes. The magnified sections in Figs. 4.50(b) and (c) clearly show a significant reduction in fine skin lines around the eyes in this case. In fact, the smoothed images look quite soft and pleasing.

We discuss unsharp masking in the frequency domain in Section 4.9.5

Figure 4.51 shows two applications of lowpass filtering on the same image, but with totally different objectives. Figure 4.51(a) is an 808×754 very high resolution radiometer (VHRR) image showing part of the Gulf of Mexico (dark) and Florida (light), taken from a NOAA satellite (note the horizontal sensor scan lines). The boundaries between bodies of water were caused by loop currents. This image is illustrative of remotely sensed images in which sensors have the tendency to produce pronounced scan lines along the direction in which the scene is being scanned (see Example 4.24 for an illustration of a



a b c

FIGURE 4.50 (a) Original image (784 × 732 pixels). (b) Result of filtering using a GLPF with $D_0 = 100$. (c) Result of filtering using a GLPF with $D_0 = 80$. Note the reduction in fine skin lines in the magnified sections in (b) and (c).

physical cause). Lowpass filtering is a crude but simple way to reduce the effect of these lines, as Fig. 4.51(b) shows (we consider more effective approaches in Sections 4.10 and 5.4.1). This image was obtained using a GLFP with $D_0 = 50$. The reduction in the effect of the scan lines can simplify the detection of features such as the interface boundaries between ocean currents.

Figure 4.51(c) shows the result of significantly more aggressive Gaussian lowpass filtering with $D_0 = 20$. Here, the objective is to blur out as much detail as possible while leaving large features recognizable. For instance, this type of filtering could be part of a preprocessing stage for an image analysis system that searches for features in an image bank. An example of such features could be lakes of a given size, such as Lake Okeechobee in the lower eastern region of Florida, shown as a nearly round dark region in Fig. 4.51(c). Lowpass filtering helps simplify the analysis by averaging out features smaller than the ones of interest.

4.9 Image Sharpening Using Frequency Domain Filters

In the previous section, we showed that an image can be smoothed by attenuating the high-frequency components of its Fourier transform. Because edges and other abrupt changes in intensities are associated with high-frequency components, image sharpening can be achieved in the frequency domain by highpass filtering, which attenuates the low-frequency components without disturbing high-frequency information in the Fourier transform. As in Section

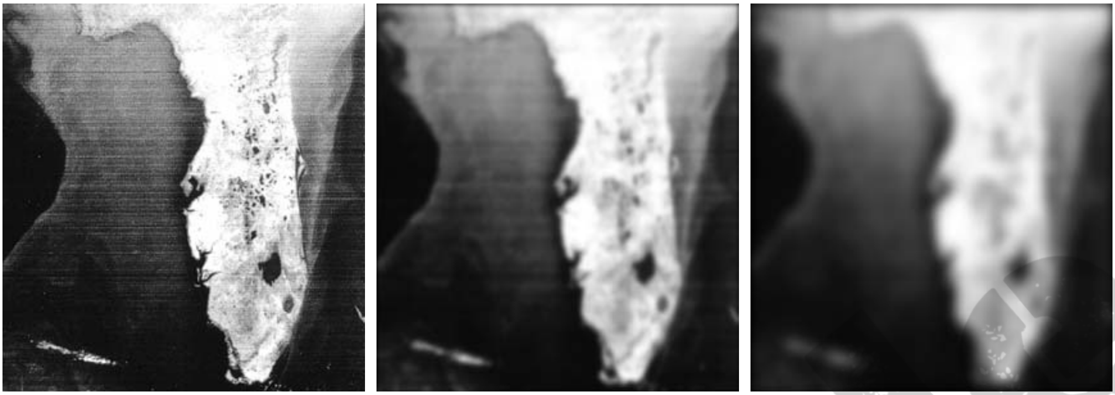


FIGURE 4.51 (a) Image showing prominent horizontal scan lines. (b) Result of filtering using a GLPF with $D_0 = 50$. (c) Result of using a GLPF with $D_0 = 20$. (Original image courtesy of NOAA.)

4.8, we consider only zero-phase-shift filters that are radially symmetric. All filtering in this section is based on the procedure outlined in Section 4.7.3, so all filter functions, $H(u, v)$, are understood to be discrete functions of size $P \times Q$; that is, the discrete frequency variables are in the range $u = 0, 1, 2, \ldots, P - 1$ and $v = 0, 1, 2, \ldots, Q - 1$.

A highpass filter is obtained from a given lowpass filter using the equation

$$H_{\text{HP}}(u, v) = 1 - H_{\text{LP}}(u, v) \tag{4.9-1}$$

where $H_{\text{LP}}(u, v)$ is the transfer function of the lowpass filter. That is, when the lowpass filter attenuates frequencies, the highpass filter passes them, and vice versa.

In this section, we consider ideal, Butterworth, and Gaussian highpass filters. As in the previous section, we illustrate the characteristics of these filters in both the frequency and spatial domains. Figure 4.52 shows typical 3-D plots, image representations, and cross sections for these filters. As before, we see that the Butterworth filter represents a transition between the sharpness of the ideal filter and the broad smoothness of the Gaussian filter. Figure 4.53, discussed in the sections that follow, illustrates what these filters look like in the spatial domain. The spatial filters were obtained and displayed by using the procedure used to generate Figs. 4.43 and 4.46.

4.9.1 Ideal Highpass Filters

A 2-D *ideal highpass filter* (IHPF) is defined as

$$H(u, v) = \begin{cases} 0 & \text{if } D(u, v) \leq D_0 \\ 1 & \text{if } D(u, v) > D_0 \end{cases} \tag{4.9-2}$$

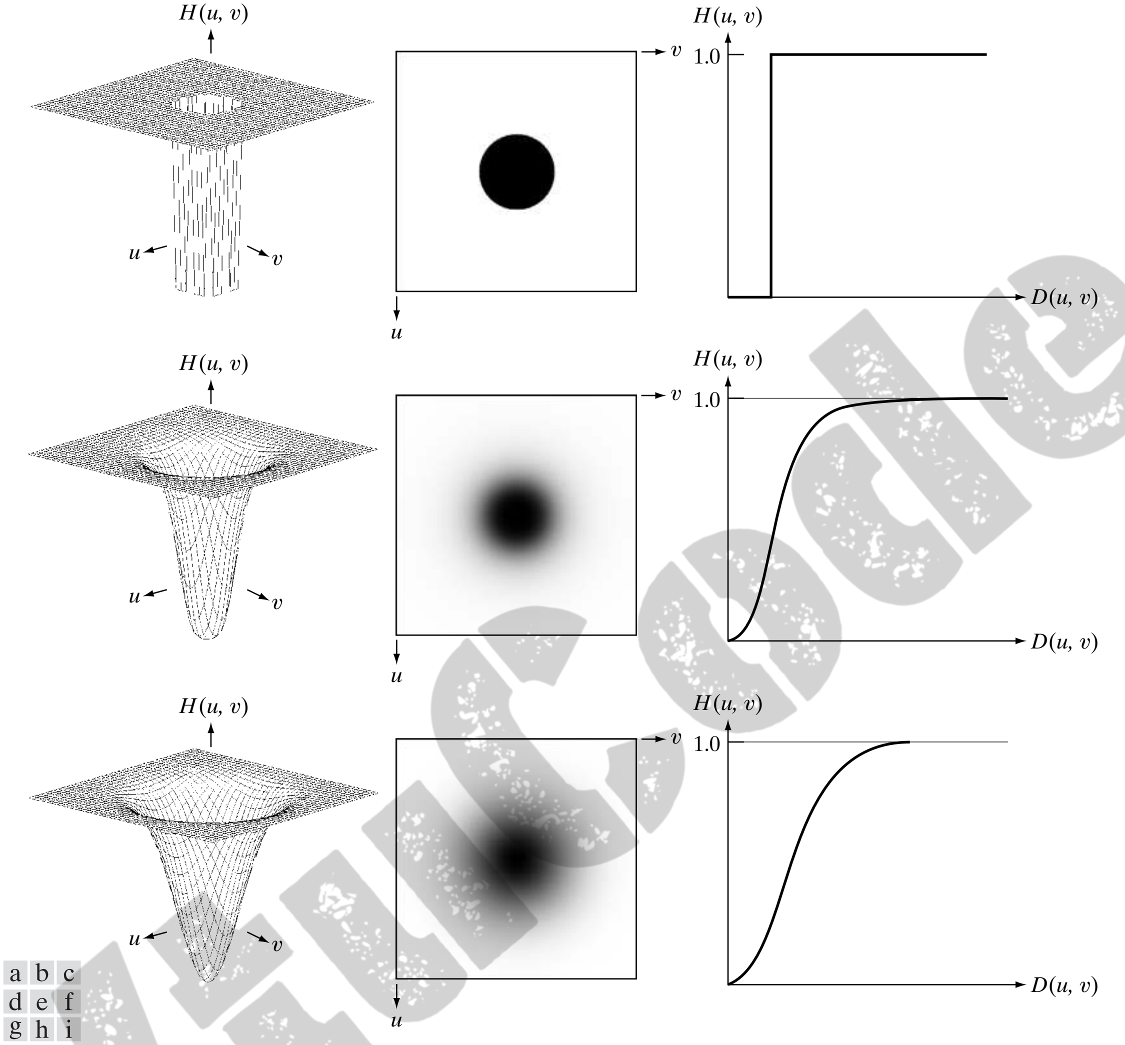


FIGURE 4.52 Top row: Perspective plot, image representation, and cross section of a typical ideal highpass filter. Middle and bottom rows: The same sequence for typical Butterworth and Gaussian highpass filters.

where D_0 is the cutoff frequency and $D(u, v)$ is given by Eq. (4.8-2). This expression follows directly from Eqs. (4.8-1) and (4.9-1). As intended, the IHPF is the opposite of the ILPF in the sense that it sets to zero all frequencies inside a circle of radius D_0 while passing, without attenuation, all frequencies outside the circle. As in the case of the ILPF, the IHPF is not physically realizable. However, we consider it here for completeness and, as before, because its properties can be used to explain phenomena such as ringing in the spatial domain. The discussion will be brief.

Because of the way in which they are related [Eq. (4.9-1)], we can expect IHPFs to have the same ringing properties as ILPFs. This is demonstrated

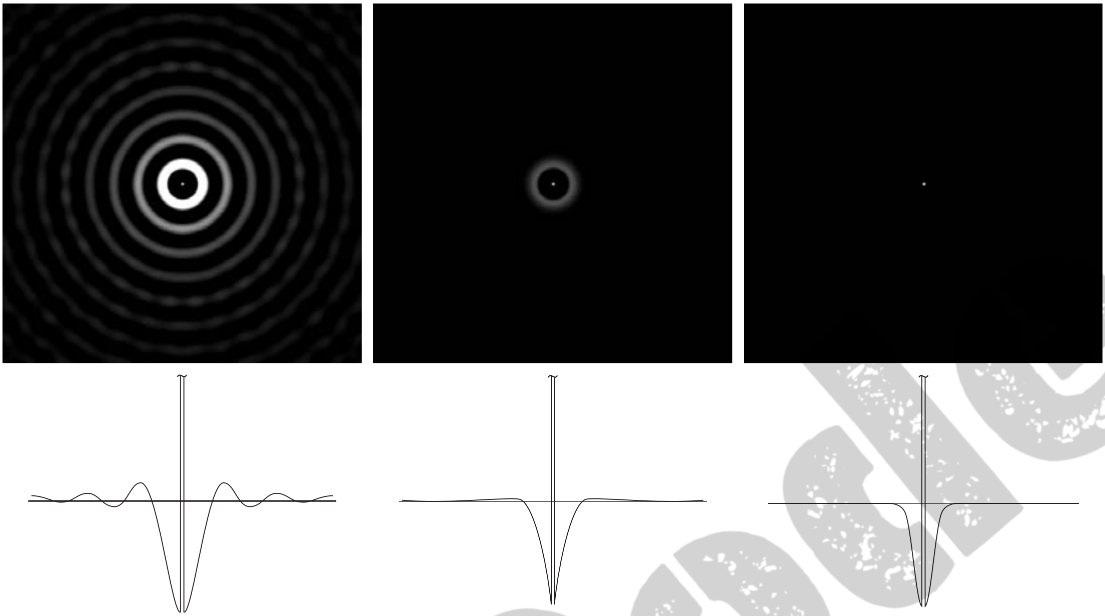


FIGURE 4.53 Spatial representation of typical (a) ideal, (b) Butterworth, and (c) Gaussian frequency domain highpass filters, and corresponding intensity profiles through their centers.

clearly in Fig. 4.54, which consists of various IHPF results using the original image in Fig. 4.41(a) with D_0 set to 30, 60, and 160 pixels, respectively. The ringing in Fig. 4.54(a) is so severe that it produced distorted, thickened object boundaries (e.g., look at the large letter "a"). Edges of the top three circles do not show well because they are not as strong as the other edges in the image (the intensity of these three objects is much closer to the background intensity,

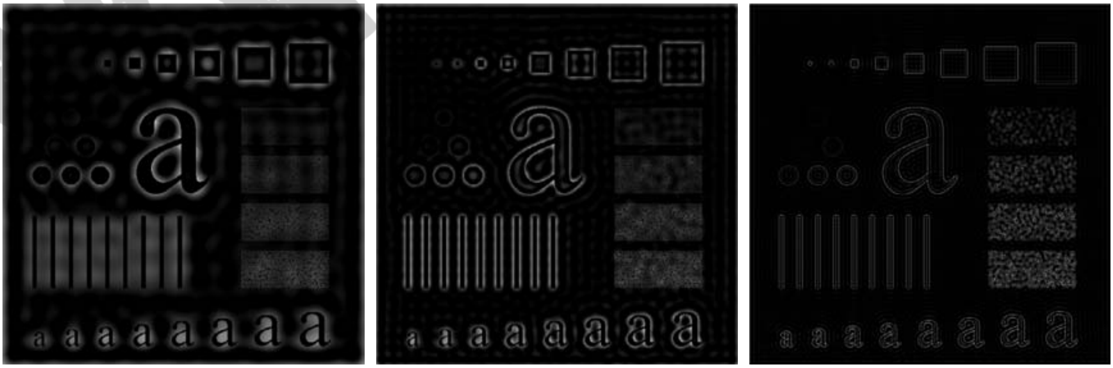


FIGURE 4.54 Results of highpass filtering the image in Fig. 4.41(a) using an IHPF with $D_0 = 30$, 60, and 160.

giving discontinuities of smaller magnitude). Looking at the "spot" size of the spatial representation of the IHPF in Fig. 4.53(a) and keeping in mind that filtering in the spatial domain is convolution of the spatial filter with the image helps explain why the smaller objects and lines appear almost solid white. Look in particular at the three small squares in the top row and the thin, vertical bars in Fig. 4.54(a). The situation improved somewhat with $D_0 = 60$. Edge distortion is quite evident still, but now we begin to see filtering on the smaller objects. Due to the now familiar inverse relationship between the frequency and spatial domains, we know that the spot size of this filter is smaller than the spot of the filter with $D_0 = 30$. The result for $D_0 = 160$ is closer to what a highpass-filtered image should look like. Here, the edges are much cleaner and less distorted, and the smaller objects have been filtered properly. Of course, the constant background in all images is zero in these highpass-filtered images because highpass filtering is analogous to differentiation in the spatial domain.

4.9.2 Butterworth Highpass Filters

A 2-D *Butterworth highpass filter* (BHPF) of order n and cutoff frequency D_0 is defined as

$$H(u, v) = \frac{1}{1 + [D_0/D(u, v)]^{2n}} \tag{4.9-3}$$

where $D(u, v)$ is given by Eq. (4.8-2). This expression follows directly from Eqs. (4.8-5) and (4.9-1). The middle row of Fig. 4.52 shows an image and cross section of the BHPF function.

As with lowpass filters, we can expect Butterworth highpass filters to behave smoother than IHPFs. Figure 4.55 shows the performance of a BHPF, of

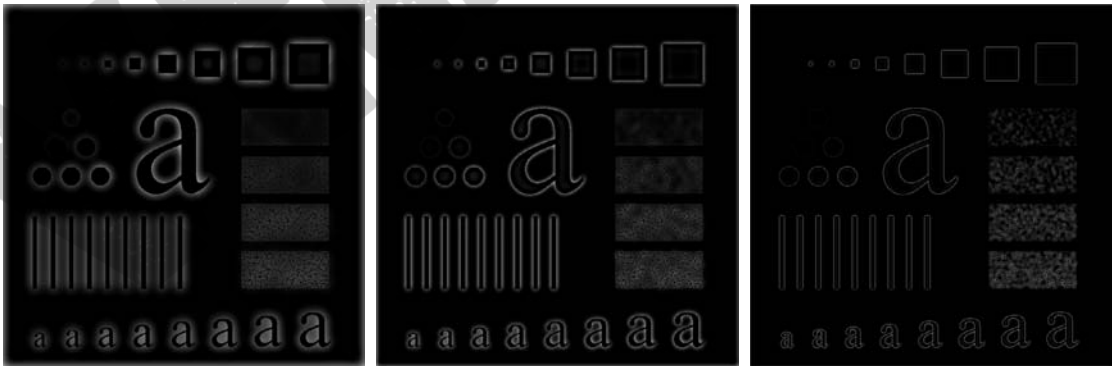


FIGURE 4.55 Results of highpass filtering the image in Fig. 4.41(a) using a BHPF of order 2 with $D_0 = 30, 60,$ and 160, corresponding to the circles in Fig. 4.41(b). These results are much smoother than those obtained with an IHPF.

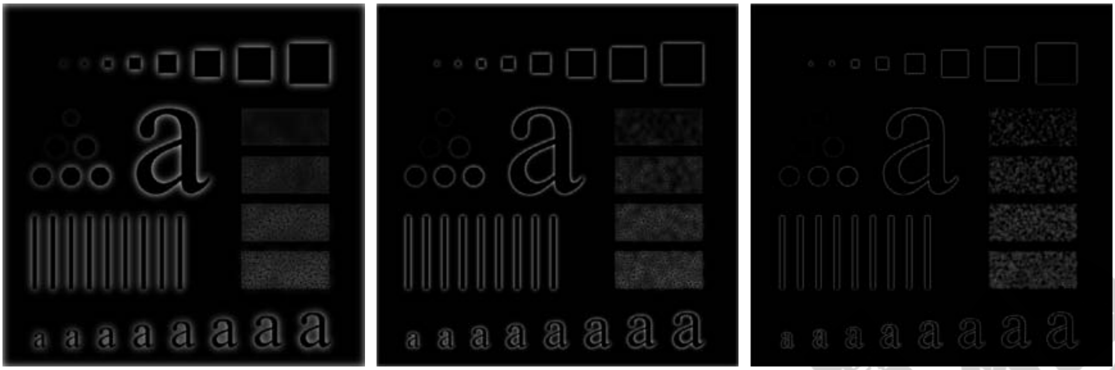


FIGURE 4.56 Results of highpass filtering the image in Fig. 4.41(a) using a GHPF with $D_0 = 30, 60$, and 160, corresponding to the circles in Fig. 4.41(b). Compare with Figs. 4.54 and 4.55.

order 2 and with D_0 set to the same values as in Fig. 4.54. The boundaries are much less distorted than in Fig. 4.54, even for the smallest value of cutoff frequency. Because the spot sizes in the center areas of the IHPF and the BHPF are similar [see Figs. 4.53(a) and (b)], the performance of the two filters on the smaller objects is comparable. The transition into higher values of cutoff frequencies is much smoother with the BHPF.

4.9.3 Gaussian Highpass Filters

The transfer function of the Gaussian highpass filter (GHPF) with cutoff frequency locus at a distance D_0 from the center of the frequency rectangle is given by

$$H(u, v) = 1 - e^{-D^2(u,v)/2D_0^2} \tag{4.9-4}$$

where $D(u, v)$ is given by Eq. (4.8-2). This expression follows directly from Eqs. (4.8-7) and (4.9-1). The third row in Fig. 4.52 shows a perspective plot, image, and cross section of the GHPF function. Following the same format as for the BHPF, we show in Fig. 4.56 comparable results using GHPFs. As expected, the results obtained are more gradual than with the previous two filters. Even the filtering of the smaller objects and thin bars is cleaner with the Gaussian filter. Table 4.5 contains a summary of the highpass filters discussed in this section.

TABLE 4.5

Highpass filters. D_0 is the cutoff frequency and n is the order of the Butterworth filter.

Ideal	Butterworth	Gaussian
$H(u,v) = \begin{cases} 1 & \text{if } D(u,v) \le D_0 \\ 0 & \text{if } D(u,v) > D_0 \end{cases}$	$H(u,v) = \dfrac{1}{1 + [D_0/D(u,v)]^{2n}}$	$H(u,v) = 1 - e^{-D^2(u,v)/2D_0^2}$

EXAMPLE 4.19: Using highpass filtering and thresholding for image enhancement.

■ Figure 4.57(a) is a 1026 × 962 image of a thumb print in which smudges (a typical problem) are evident. A key step in automated fingerprint recognition is enhancement of print ridges and the reduction of smudges. Enhancement is useful also in human interpretation of prints. In this example, we use highpass filtering to enhance the ridges and reduce the effects of smudging. Enhancement of the ridges is accomplished by the fact that they contain high frequencies, which are unchanged by a highpass filter. On the other hand, the filter reduces low frequency components, which correspond to slowly varying intensities in the image, such as the background and smudges. Thus, enhancement is achieved by reducing the effect of all features except those with high frequencies, which are the features of interest in this case.

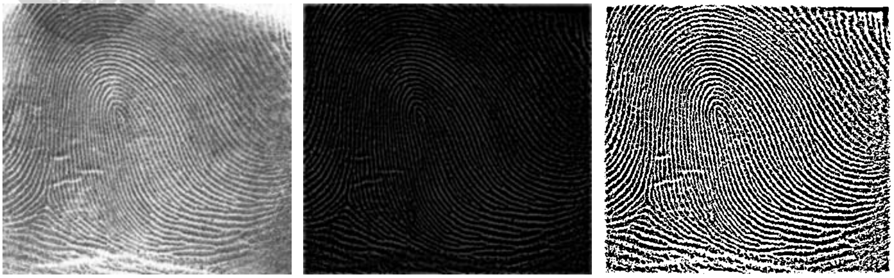
Figure 4.57(b) is the result of using a Butterworth highpass filter of order 4 with a cutoff frequency of 50. As expected, the highpass-filtered image lost its gray tones because the dc term was reduced to 0. The net result is that dark tones typically predominate in highpass-filtered images, thus requiring additional processing to enhance details of interest. A simple approach is to threshold the filtered image. Figure 4.57(c) shows the result of setting to black all negative values and to white all positive values in the filtered image. Note how the ridges are clear and the effect of the smudges has been reduced considerably. In fact, ridges that are barely visible in the top, right section of the image in Fig. 4.57(a) are nicely enhanced in Fig. 4.57(c). ■

The value $D_0 = 50$ is approximately 2.5% of the short dimension of the padded image. The idea is for D_0 to be close to the origin so low frequencies are attenuated, but not completely eliminated. A range of 2% to 5% of the short dimension is a good starting point.

4.9.4 The Laplacian in the Frequency Domain

In Section 3.6.2, we used the Laplacian for image enhancement in the spatial domain. In this section, we revisit the Laplacian and show that it yields equivalent results using frequency domain techniques. It can be shown (Problem 4.26) that the Laplacian can be implemented in the frequency domain using the filter

$$H(u, v) = -4\pi^2(u^2 + v^2) \tag{4.9-5}$$



a b c

FIGURE 4.57 (a) Thumb print. (b) Result of highpass filtering (a). (c) Result of thresholding (b). (Original image courtesy of the U.S. National Institute of Standards and Technology.)

or, with respect to the center of the frequency rectangle, using the filter

$$\begin{aligned} H(u, v) &= -4\pi^2\left[(u - P/2)^2 + (v - Q/2)^2\right] \\ &= -4\pi^2 D^2(u, v) \end{aligned} \tag{4.9-6}$$

where $D(u, v)$ is the distance function given in Eq. (4.8-2). Then, the Laplacian image is obtained as:

$$\nabla^2 f(x, y) = \Im^{-1}\left\{H(u, v)F(u, v)\right\} \tag{4.9-7}$$

where $F(u, v)$ is the DFT of $f(x, y)$. As explained in Section 3.6.2, enhancement is achieved using the equation:

$$g(x, y) = f(x, y) + c\nabla^2 f(x, y) \tag{4.9-8}$$

Here, $c = -1$ because $H(u, v)$ is negative. In Chapter 3, $f(x, y)$ and $\nabla^2 f(x, y)$ had comparable values. However, computing $\nabla^2 f(x, y)$ with Eq. (4.9-7) introduces DFT scaling factors that can be several orders of magnitude larger than the maximum value of f. Thus, the differences between f and its Laplacian must be brought into comparable ranges. The easiest way to handle this problem is to normalize the values of $f(x, y)$ to the range [0, 1] (before computing its DFT) and divide $\nabla^2 f(x, y)$ by its maximum value, which will bring it to the approximate range $[-1, 1]$ (recall that the Laplacian has negative values). Equation (4.9-8) can then be applied.

In the frequency domain, Eq. (4.9-8) is written as

$$\begin{aligned} g(x, y) &= \Im^{-1}\left\{F(u, v) - H(u, v)F(u, v)\right\} \\ &= \Im^{-1}\left\{\left[1 - H(u, v)\right]F(u, v)\right\} \\ &= \Im^{-1}\left\{\left[1 + 4\pi^2 D^2(u, v)\right]F(u, v)\right\} \end{aligned} \tag{4.9-9}$$

Although this result is elegant, it has the same scaling issues just mentioned, compounded by the fact that the normalizing factor is not as easily computed. For this reason, Eq. (4.9-8) is the preferred implementation in the frequency domain, with $\nabla^2 f(x, y)$ computed using Eq. (4.9-7) and scaled using the approach mentioned in the previous paragraph.

EXAMPLE 4.20: Image sharpening in the frequency domain using the Laplacian.

■ Figure 4.58(a) is the same as Fig. 3.38(a), and Fig. 4.58(b) shows the result of using Eq. (4.9-8), in which the Laplacian was computed in the frequency domain using Eq. (4.9-7). Scaling was done as described in connection with that equation. We see by comparing Figs. 4.58(b) and 3.38(e) that the frequency domain and spatial results are identical visually. Observe that the results in these two figures correspond to the Laplacian mask in Fig. 3.37(b), which has a −8 in the center (Problem 4.26). ■

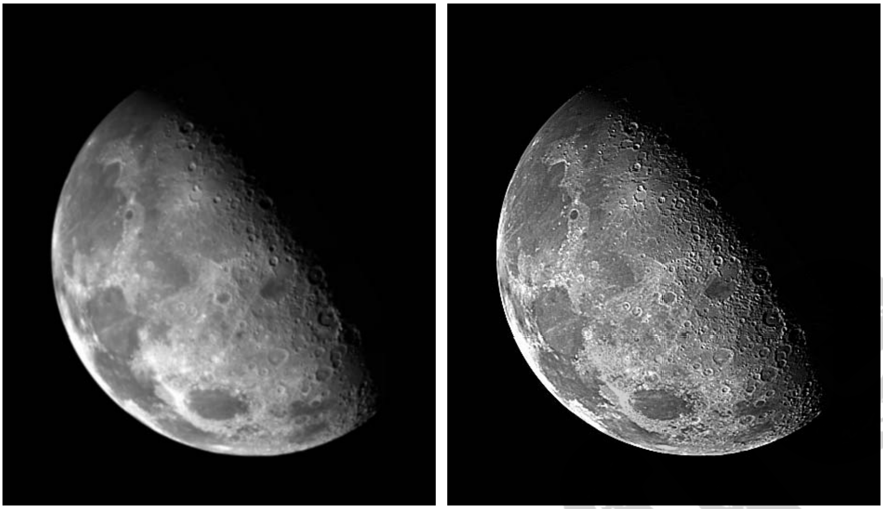


FIGURE 4.58 (a) Original, blurry image. (b) Image enhanced using the Laplacian in the frequency domain. Compare with Fig. 3.38(e).

4.9.5 Unsharp Masking, Highboost Filtering, and High-Frequency-Emphasis Filtering

In this section, we discuss frequency domain formulations of the unsharp masking and high-boost filtering image sharpening techniques introduced in Section 3.6.3. Using frequency domain methods, the mask defined in Eq. (3.6-8) is given by

$$g_{\text{mask}}(x, y) = f(x, y) - f_{\text{LP}}(x, y) \tag{4.9-10}$$

with

$$f_{\text{LP}}(x, y) = \mathfrak{I}^{-1}\left[H_{\text{LP}}(u, v)F(u, v)\right] \tag{4.9-11}$$

where $H_{\text{LP}}(u, v)$ is a lowpass filter and $F(u, v)$ is the Fourier transform of $f(x, y)$. Here, $f_{\text{LP}}(x, y)$ is a smoothed image analogous to $\bar{f}(x, y)$ in Eq. (3.6-8). Then, as in Eq. (3.6-9),

$$g(x, y) = f(x, y) + k * g_{\text{mask}}(x, y) \tag{4.9-12}$$

This expression defines unsharp masking when $k = 1$ and highboost filtering when $k > 1$. Using the preceding results, we can express Eq. (4.9-12) entirely in terms of frequency domain computations involving a lowpass filter:

$$g(x, y) = \mathfrak{I}^{-1}\left\{\left[1 + k * [1 - H_{\text{LP}}(u, v)]\right]F(u, v)\right\} \tag{4.9-13}$$

Using Eq. (4.9-1), we can express this result in terms of a highpass filter:

$$g(x, y) = \mathfrak{I}^{-1}\left\{[1 + k * H_{\text{HP}}(u, v)]F(u, v)\right\} \tag{4.9-14}$$

The expression contained within the square brackets is called a *high-frequency-emphasis filter*. As noted earlier, highpass filters set the dc term to zero, thus reducing the average intensity in the filtered image to 0. The high-frequency-emphasis filter does not have this problem because of the 1 that is added to the highpass filter. The constant, k, gives control over the proportion of high frequencies that influence the final result. A slightly more general formulation of high-frequency-emphasis filtering is the expression

$$g(x, y) = \Im^{-1}\left\{[k_1 + k_2 * H_{\text{HP}}(u, v)]F(u, v)\right\} \tag{4.9-15}$$

where $k_1 \geq 0$ gives controls of the offset from the origin [see Fig. 4.31(c)] and $k_2 \geq 0$ controls the contribution of high frequencies.

EXAMPLE 4.21: Image enhancement using high-frequency-emphasis filtering.

■ Figure 4.59(a) shows a 416 × 596 chest X-ray with a narrow range of intensity levels. The objective of this example is to enhance the image using high-frequency-emphasis filtering. X-rays cannot be focused in the same manner that optical lenses are focused, and the resulting images generally tend to be slightly blurred. Because the intensities in this particular image are biased toward the dark end of the gray scale, we also take this opportunity to give an example of how spatial domain processing can be used to complement frequency-domain filtering.

Figure 4.59(b) shows the result of highpass filtering using a Gaussian filter with $D_0 = 40$ (approximately 5% of the short dimension of the padded image). As expected, the filtered result is rather featureless, but it shows faintly the principal edges in the image. Figure 4.59(c) shows the advantage of high-emphasis filtering, where we used Eq. (4.9-15) with $k_1 = 0.5$ and $k_2 = 0.75$. Although the image is still dark, the gray-level tonality due to the low-frequency components was not lost.

Artifacts such as ringing are unacceptable in medical imaging. Thus, it is good practice to avoid using filters that have the potential for introducing artifacts in the processed image. Because spatial and frequency domain Gaussian filters are Fourier transform pairs, these filters produce smooth results that are void of artifacts.

As discussed in Section 3.3.1, an image characterized by intensity levels in a narrow range of the gray scale is an ideal candidate for histogram equalization. As Fig. 4.59(d) shows, this was indeed an appropriate method to further enhance the image. Note the clarity of the bone structure and other details that simply are not visible in any of the other three images. The final enhanced image is a little noisy, but this is typical of X-ray images when their gray scale is expanded. The result obtained using a combination of high-frequency emphasis and histogram equalization is superior to the result that would be obtained by using either method alone. ■

4.9.6 Homomorphic Filtering

The illumination-reflectance model introduced in Section 2.3.4 can be used to develop a frequency domain procedure for improving the appearance of an image by simultaneous intensity range compression and contrast enhancement. From the discussion in that section, an image $f(x, y)$ can be expressed as the product of its illumination, $i(x, y)$, and reflectance, $r(x, y)$, components:

$$f(x, y) = i(x, y)r(x, y) \tag{4.9-16}$$

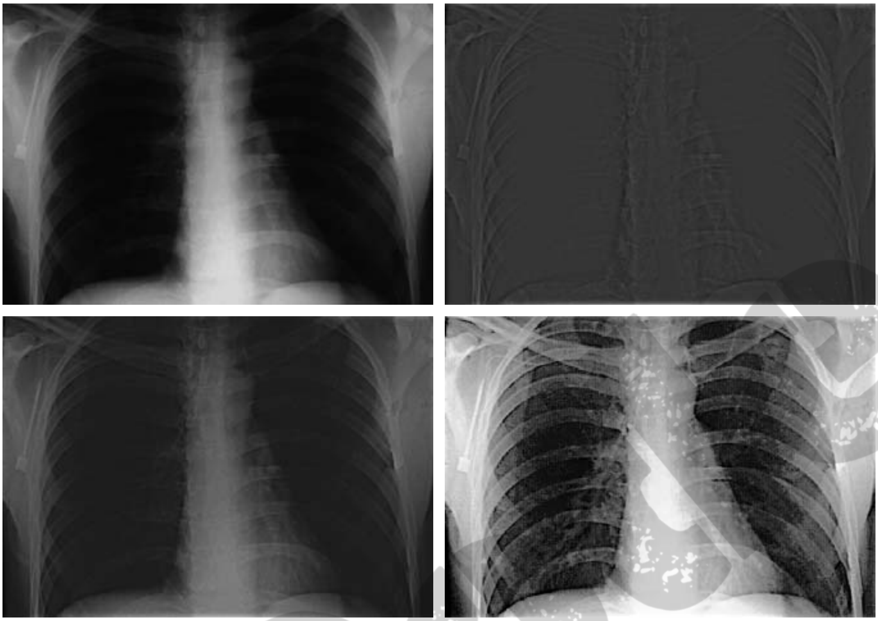


FIGURE 4.59 (a) A chest X-ray image. (b) Result of highpass filtering with a Gaussian filter. (c) Result of high-frequency-emphasis filtering using the same filter. (d) Result of performing histogram equalization on (c). (Original image courtesy of Dr. Thomas R. Gest, Division of Anatomical Sciences, University of Michigan Medical School.)

This equation cannot be used directly to operate on the frequency components of illumination and reflectance because the Fourier transform of a product is not the product of the transforms:

$$\Im[f(x, y)] \neq \Im[i(x, y)]\Im[r(x, y)] \tag{4.9-17}$$

However, suppose that we define

If an image $f(x, y)$ with intensities in the range $[0, L - 1]$ has any 0 values, a 1 must be added to every element of the image to avoid having to deal with ln(0). The 1 is then subtracted at the end of the filtering process.

$$\begin{aligned} z(x, y) &= \ln f(x, y) \\ &= \ln i(x, y) + \ln r(x, y) \end{aligned} \tag{4.9-18}$$

Then,

$$\begin{aligned} \Im\{z(x, y)\} &= \Im\{\ln f(x, y)\} \\ &= \Im\{\ln i(x, y)\} + \Im\{\ln r(x, y)\} \end{aligned} \tag{4.9-19}$$

or

$$Z(u, v) = F_i(u, v) + F_r(u, v) \tag{4.9-20}$$

where $F_i(u, v)$ and $F_r(u, v)$ are the Fourier transforms of $\ln i(x, y)$ and $\ln r(x, y)$, respectively.

We can filter $Z(u, v)$ using a filter $H(u, v)$ so that

$$\begin{aligned} S(u, v) &= H(u, v)Z(u, v) \\ &= H(u, v)F_i(u, v) + H(u, v)F_r(u, v) \end{aligned} \tag{4.9-21}$$

The filtered image in the spatial domain is

$$\begin{aligned} s(x, y) &= \Im^{-1}\{S(u, v)\} \\ &= \Im^{-1}\{H(u, v)F_i(u, v)\} + \Im^{-1}\{H(u, v)F_r(u, v)\} \end{aligned} \tag{4.9-22}$$

By defining

$$i'(x, y) = \Im^{-1}\{H(u, v)F_i(u, v)\} \tag{4.9-23}$$

and

$$r'(x, y) = \Im^{-1}\{H(u, v)F_r(u, v)\} \tag{4.9-24}$$

we can express Eq. (4.9-23) in the form

$$s(x, y) = i'(x, y) + r'(x, y) \tag{4.9-25}$$

Finally, because $z(x, y)$ was formed by taking the natural logarithm of the input image, we reverse the process by taking the exponential of the filtered result to form the output image:

$$\begin{aligned} g(x, y) &= e^{s(x,y)} \\ &= e^{i'(x,y)}e^{r'(x,y)} \\ &= i_0(x, y)\,r_0(x, y) \end{aligned} \tag{4.9-26}$$

where

$$i_0(x, y) = e^{i'(x,y)} \tag{4.9-27}$$

and

$$r_0(x, y) = e^{r'(x,y)} \tag{4.9-28}$$

are the illumination and reflectance components of the output (processed) image.

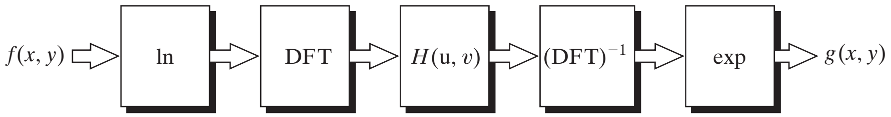


FIGURE 4.60 Summary of steps in homomorphic filtering.

The filtering approach just derived is summarized in Fig. 4.60. This method is based on a special case of a class of systems known as *homomorphic systems*. In this particular application, the key to the approach is the separation of the illumination and reflectance components achieved in the form shown in Eq. (4.9-20). The *homomorphic filter function* $H(u, v)$ then can operate on these components separately, as indicated by Eq. (4.9-21).

The illumination component of an image generally is characterized by slow spatial variations, while the reflectance component tends to vary abruptly, particularly at the junctions of dissimilar objects. These characteristics lead to associating the low frequencies of the Fourier transform of the logarithm of an image with illumination and the high frequencies with reflectance. Although these associations are rough approximations, they can be used to advantage in image filtering, as illustrated in Example 4.22.

A good deal of control can be gained over the illumination and reflectance components with a homomorphic filter. This control requires specification of a filter function $H(u, v)$ that affects the low- and high-frequency components of the Fourier transform in different, controllable ways. Figure 4.61 shows a cross section of such a filter. If the parameters γ_L and γ_H are chosen so that $\gamma_L < 1$ and $\gamma_H > 1$, the filter function in Fig. 4.61 tends to attenuate the contribution made by the low frequencies (illumination) and amplify the contribution made by high frequencies (reflectance). The net result is simultaneous dynamic range compression and contrast enhancement.

The shape of the function in Fig. 4.61 can be approximated using the basic form of a highpass filter. For example, using a slightly modified form of the Gaussian highpass filter yields the function

$$H(u, v) = (\gamma_H - \gamma_L)\left[1 - e^{-c[D^2(u, v)/D_0^2]}\right] + \gamma_L \qquad (4.9\text{-}29)$$

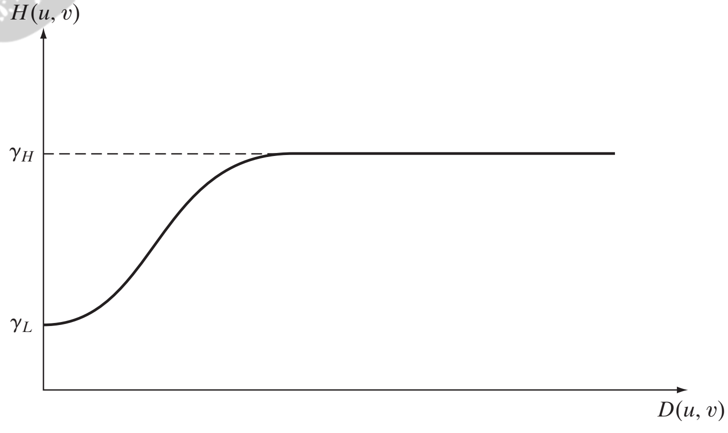


FIGURE 4.61 Radial cross section of a circularly symmetric homomorphic filter function. The vertical axis is at the center of the frequency rectangle and $D(u, v)$ is the distance from the center.

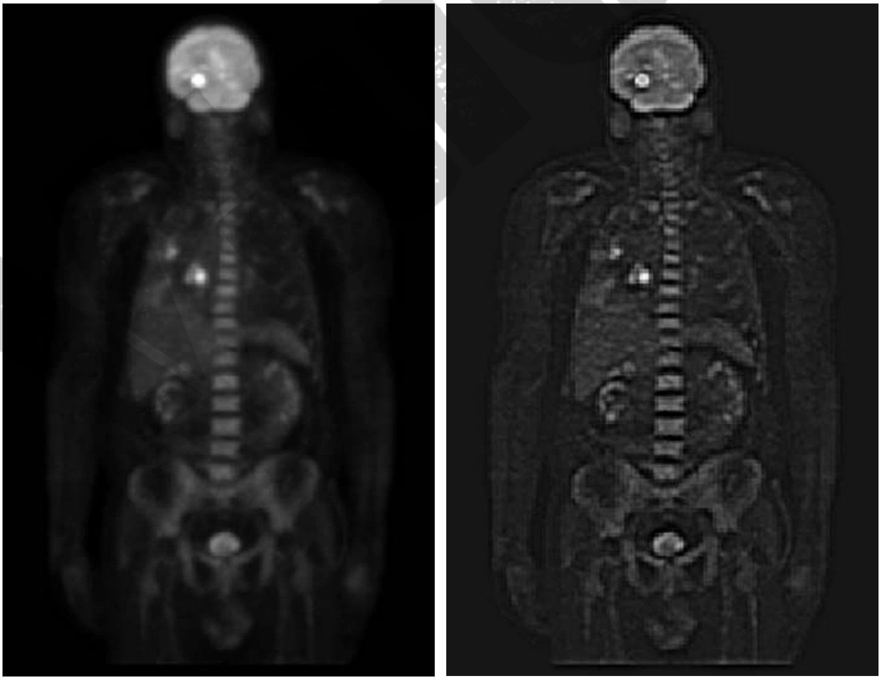
where $D(u, v)$ is defined in Eq. (4.8-2) and the constant c controls the sharpness of the slope of the function as it transitions between γ_L and γ_H. This filter is similar to the high-emphasis filter discussed in the previous section.

EXAMPLE 4.22: Image enhancement using homomorphic filtering.

■ Figure 4.62(a) shows a full body PET (Positron Emission Tomography) scan of size 1162 × 746 pixels. The image is slightly blurry and many of its low-intensity features are obscured by the high intensity of the "hot spots" dominating the dynamic range of the display. (These hot spots were caused by a tumor in the brain and one in the lungs.) Figure 4.62(b) was obtained by homomorphic filtering Fig. 4.62(a) using the filter in Eq. (4.9-29) with $\gamma_L = 0.25, \gamma_H = 2, c = 1$, and $D_0 = 80$. A cross section of this filter looks just like Fig. 4.61, with a slightly steeper slope.

Recall that filtering uses image padding, so the filter is of size $P \times Q$.

Note in Fig. 4.62(b) how much sharper the hot spots, the brain, and the skeleton are in the processed image, and how much more detail is visible in this image. By reducing the effects of the dominant illumination components (the hot spots), it became possible for the dynamic range of the display to allow lower intensities to become much more visible. Similarly, because the high frequencies are enhanced by homomorphic filtering, the reflectance components of the image (edge information) were sharpened considerably. The enhanced image in Fig. 4.62(b) is a significant improvement over the original. ■



a b

FIGURE 4.62 (a) Full body PET scan. (b) Image enhanced using homomorphic filtering. (Original image courtesy of Dr. Michael E. Casey, CTI PET Systems.)

4.10 Selective Filtering

The filters discussed in the previous two sections operate over the entire frequency rectangle. There are applications in which it is of interest to process specific bands of frequencies or small regions of the frequency rectangle. Filters in the first category are called *bandreject* or *bandpass filters*, respectively. Filters in the second category are called *notch filters*.

4.10.1 Bandreject and Bandpass Filters

These types of filters are easy to construct using the concepts from the previous two sections. Table 4.6 shows expressions for ideal, Butterworth, and Gaussian bandreject filters, where $D(u, v)$ is the distance from the center of the frequency rectangle, as given in Eq. (4.8-2), D_0 is the radial center of the band, and W is the width of the band. Figure 4.63(a) shows a Gaussian bandreject filter in image form, where black is 0 and white is 1.

A bandpass filter is obtained from a bandreject filter in the same manner that we obtained a highpass filter from a lowpass filter:

$$H_{\text{BP}}(u, v) = 1 - H_{\text{BR}}(u, v) \tag{4.10-1}$$

Figure 4.63(b) shows a Gaussian bandpass filter in image form.

4.10.2 Notch Filters

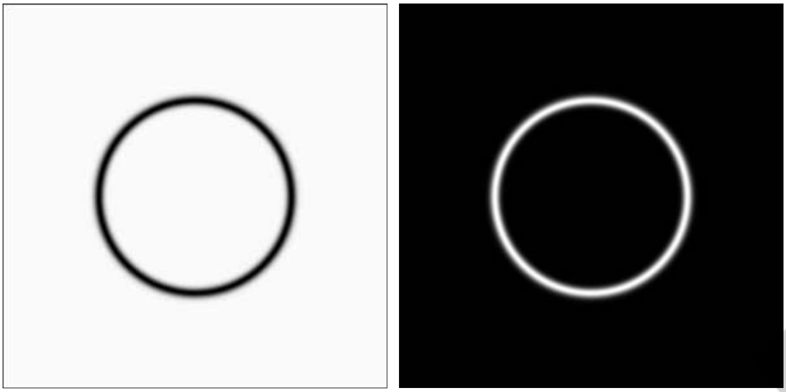
Notch filters are the most useful of the selective filters. A notch filter rejects (or passes) frequencies in a predefined neighborhood about the center of the frequency rectangle. Zero-phase-shift filters must be symmetric about the origin, so a notch with center at (u_0, v_0) must have a corresponding notch at location $(-u_0, -v_0)$. Notch reject filters are constructed as products of highpass filters whose centers have been translated to the centers of the notches. The general form is:

$$H_{\text{NR}}(u, v) = \prod_{k=1}^{Q} H_k(u, v) H_{-k}(u, v) \tag{4.10-2}$$

where $H_k(u, v)$ and $H_{-k}(u, v)$ are highpass filters whose centers are at (u_k, v_k) and $(-u_k, -v_k)$, respectively. These centers are specified with respect to the

TABLE 4.6
Bandreject filters. W is the width of the band, D is the distance $D(u, v)$ from the center of the filter, D_0 is the cutoff frequency, and n is the order of the Butterworth filter. We show D instead of $D(u, v)$ to simplify the notation in the table.

Ideal	Butterworth	Gaussian
$H(u, v) = \begin{cases} 0 & \text{if } D_0 - \dfrac{W}{2} \leq D \leq D_0 + \dfrac{W}{2} \\ 1 & \text{otherwise} \end{cases}$	$H(u, v) = \dfrac{1}{1 + \left[\dfrac{DW}{D^2 - D_0^2}\right]^{2n}}$	$H(u, v) = 1 - e^{-\left[\frac{D^2 - D_0^2}{DW}\right]^2}$



a b

FIGURE 4.63 (a) Bandreject Gaussian filter. (b) Corresponding bandpass filter. The thin black border in (a) was added for clarity; it is not part of the data.

center of the frequency rectangle, (M/2, N/2). The distance computations for each filter are thus carried out using the expressions

$$D_k(u, v) = \left[(u - M/2 - u_k)^2 + (v - N/2 - v_k)^2\right]^{1/2} \qquad (4.10\text{-}3)$$

and

$$D_{-k}(u, v) = \left[(u - M/2 + u_k)^2 + (v - N/2 + v_k)^2\right]^{1/2} \qquad (4.10\text{-}4)$$

For example, the following is a Butterworth notch reject filter of order n, containing three notch pairs:

$$H_{\mathrm{NR}}(u, v) = \prod_{k=1}^{3}\left[\frac{1}{1 + [D_{0k}/D_k(u, v)]^{2n}}\right]\left[\frac{1}{1 + [D_{0k}/D_{-k}(u, v)]^{2n}}\right] \qquad (4.10\text{-}5)$$

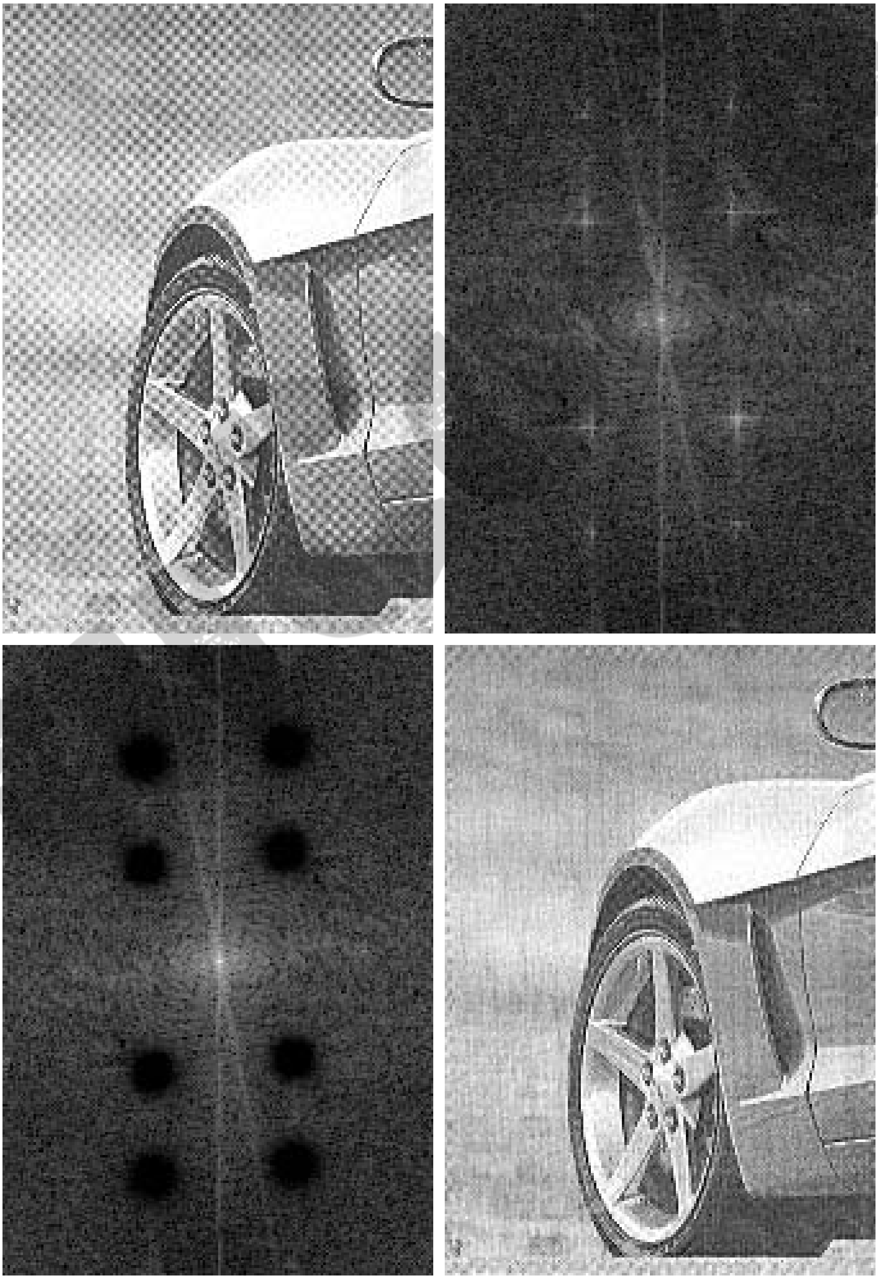
where D_k and D_{-k} are given by Eqs. (4.10-3) and (4.10-4). The constant D_{0k} is the same for each pair of notches, but it can be different for different pairs. Other notch reject filters are constructed in the same manner, depending on the highpass filter chosen. As with the filters discussed earlier, a *notch pass filter* is obtained from a notch reject filter using the expression

$$H_{\mathrm{NP}}(u, v) = 1 - H_{\mathrm{NR}}(u, v) \qquad (4.10\text{-}6)$$

As the next three examples show, one of the principal applications of notch filtering is for selectively modifying local regions of the DFT. This type of processing typically is done interactively, working directly on DFTs obtained without padding. The advantages of working interactively with actual DFTs (as opposed to having to "translate" from padded to actual frequency values) outweigh any wraparound errors that may result from not using padding in the filtering process. Also, as we show in Section 5.4.4, even more powerful notch filtering techniques than those discussed here are based on unpadded DFTs. To get an idea of how DFT values change as a function of padding, see Problem 4.22.

EXAMPLE 4.23:
Reduction of moiré patterns using notch filtering.

■ Figure 4.64(a) is the scanned newspaper image from Fig. 4.21, showing a prominent moiré pattern, and Fig. 4.64(b) is its spectrum. We know from Table 4.3 that the Fourier transform of a pure sine, which is a periodic function, is a pair of conjugate symmetric impulses. The symmetric "impulse-like" bursts in Fig. 4.64(b) are a result of the near periodicity of the moiré pattern. We can attenuate these bursts by using notch filtering.



a b
c d

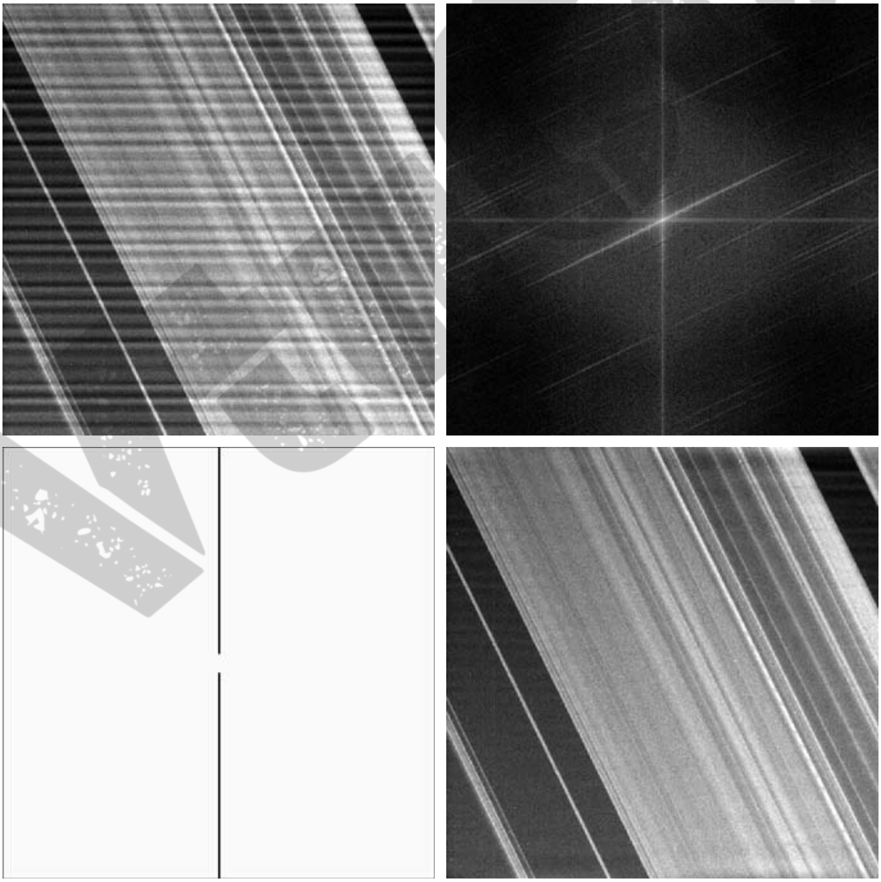
FIGURE 4.64
(a) Sampled newspaper image showing a moiré pattern. (b) Spectrum. (c) Butterworth notch reject filter multiplied by the Fourier transform. (d) Filtered image.

Figure 4.64(c) shows the result of multiplying the DFT of Fig. 4.64(a) by a Butterworth notch reject filter with $D_0 = 3$ and $n = 4$ for all notch pairs. The value of the radius was selected (by visual inspection of the spectrum) to encompass the energy bursts completely, and the value of n was selected to give notches with mildly sharp transitions. The locations of the center of the notches were determined interactively from the spectrum. Figure 4.64(d) shows the result obtained with this filter using the procedure outlined in Section 4.7.3. The improvement is significant, considering the low resolution and degradation of the original image. ■

EXAMPLE 4.24: Enhancement of corrupted *Cassini* Saturn image by notch filtering.

■ Figure 4.65(a) shows an image of part of the rings surrounding the planet Saturn. This image was captured by *Cassini*, the first spacecraft to enter the planet's orbit. The vertical sinusoidal pattern was caused by an AC signal superimposed on the camera video signal just prior to digitizing the image. This was an unexpected problem that corrupted some images from the mission. Fortunately, this type of interference is fairly easy to correct by postprocessing. One approach is to use notch filtering.

Figure 4.65(b) shows the DFT spectrum. Careful analysis of the vertical axis reveals a series of small bursts of energy which correspond to the nearly sinusoidal



a b
c d

FIGURE 4.65 (a) 674 × 674 image of the Saturn rings showing nearly periodic interference. (b) Spectrum: The bursts of energy in the vertical axis near the origin correspond to the interference pattern. (c) A vertical notch reject filter. (d) Result of filtering. The thin black border in (c) was added for clarity; it is not part of the data. (Original image courtesy of Dr. Robert A. West, NASA/JPL.)

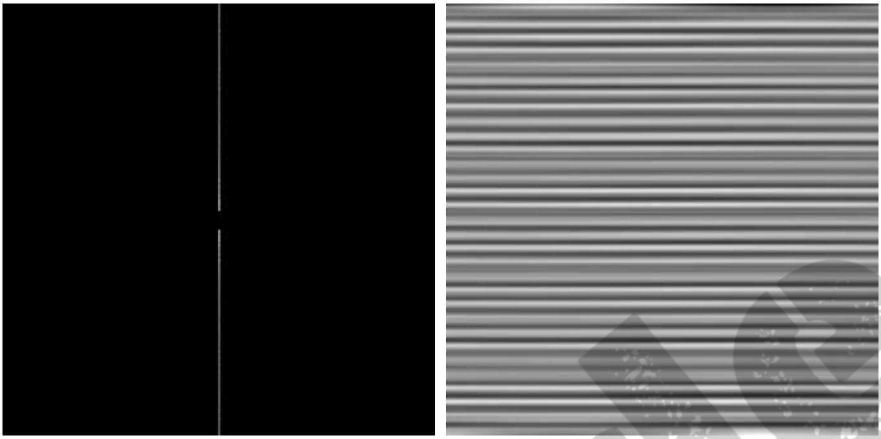


FIGURE 4.66 (a) Result (spectrum) of applying a notch pass filter to the DFT of Fig. 4.65(a). (b) Spatial pattern obtained by computing the IDFT of (a).

interference. A simple approach is to use a narrow notch rectangle filter starting with the lowest frequency burst and extending for the remaining of the vertical axis. Figure 4.65(c) shows such a filter (white represents 1 and black 0). Figure 4.65(d) shows the result of filtering the corrupted image with this filter. This result is a significant improvement over the original image.

We isolated the frequencies in the vertical axis using a notch *pass* version of the same filter [Fig. 4.66(a)]. Then, as Fig. 4.66(b) shows, the IDFT of these frequencies yielded the spatial interference pattern itself. ■