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,

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

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:

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

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.

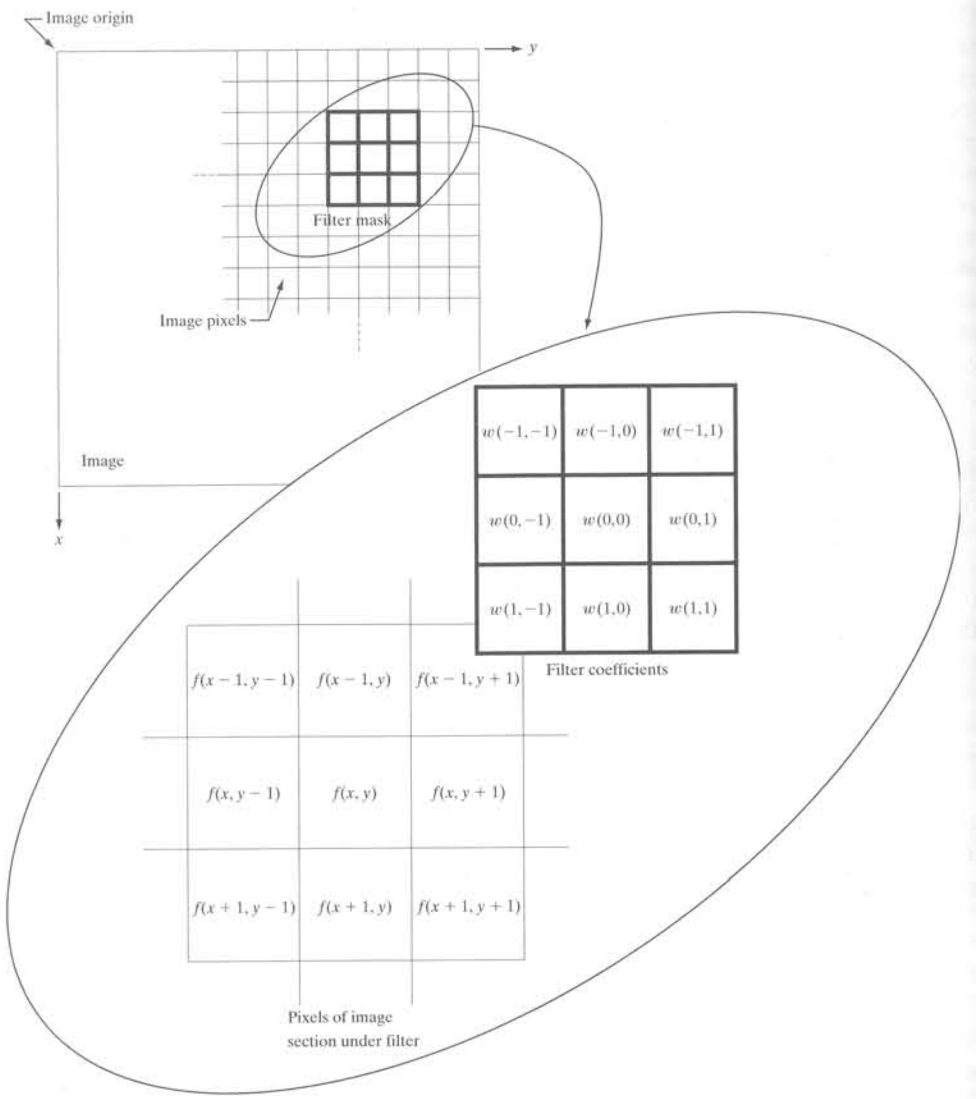


FIGURE 3.28 The mechanics of linear spatial filtering using a 3×3 filter mask. The form chosen to denote the coordinates of the filter mask coefficients simplifies writing expressions for linear filtering.

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                   Origin      f             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                                     0 0 0 1 0 0 0 0                     (j)
    1 2 3 2 8                                           8 2 3 2 1
            Starting position alignment

                   Zero padding
(c) 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0                     0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 (k)
    1 2 3 2 8                                           8 2 3 2 1

(d) 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0                     0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 (l)
      1 2 3 2 8                                           8 2 3 2 1
      Position after one shift

(e) 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0                     0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 (m)
            1 2 3 2 8                                           8 2 3 2 1
            Position after four shifts

(f) 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0                     0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 (n)
                          1 2 3 2 8                                           8 2 3 2 1
              Final position

                Full correlation result                            Full convolution result
(g)           0 0 0 8 2 3 2 1 0 0 0 0                          0 0 0 1 2 3 2 8 0 0 0 0          (o)

                Cropped correlation result                         Cropped convolution result
(h)              0 8 2 3 2 1 0 0                                  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:

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

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.

Origin $f(x, y)$

$$\begin{array}{ccccc}0&0&0&0&0\\0&0&0&0&0\\0&0&1&0&0\\0&0&0&0&0\\0&0&0&0&0\end{array} \qquad w(x,y)\quad \begin{array}{ccc}1&2&3\\4&5&6\\7&8&9\end{array}$$

(a)

Padded f

$$\begin{array}{ccccccccc}0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&1&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\end{array}$$

(b)

Initial position for w / Full correlation result / Cropped correlation result

$$\begin{array}{ccccccccc}1&2&3&0&0&0&0&0&0\\4&5&6&0&0&0&0&0&0\\7&8&9&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&1&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\end{array} \qquad \begin{array}{ccccccccc}0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&9&8&7&0&0&0\\0&0&0&6&5&4&0&0&0\\0&0&0&3&2&1&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\end{array} \qquad \begin{array}{ccccc}0&0&0&0&0\\0&9&8&7&0\\0&6&5&4&0\\0&3&2&1&0\\0&0&0&0&0\end{array}$$

(c) (d) (e)

Rotated w / Full convolution result / Cropped convolution result

$$\begin{array}{ccccccccc}9&8&7&0&0&0&0&0&0\\6&5&4&0&0&0&0&0&0\\3&2&1&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&1&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\end{array} \qquad \begin{array}{ccccccccc}0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&1&2&3&0&0&0\\0&0&0&4&5&6&0&0&0\\0&0&0&7&8&9&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\end{array} \qquad \begin{array}{ccccc}0&0&0&0&0\\0&1&2&3&0\\0&4&5&6&0\\0&7&8&9&0\\0&0&0&0&0\end{array}$$

(f) (g) (h)

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

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

Often, when the meaning is clear, we denote the result of correlation or convolution by a function $g(x, y)$, instead of writing $w(x, y) \,\text{☆}\, f(x, y)$ 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).

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

†Because correlation and convolution are commutative, we have that $w(x, y) \,\text{☆}\, f(x, y) = f(x, y) \,\text{☆}\, w(x, y)$ 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

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

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

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

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

3.4.4 Generating Spatial Filter Masks

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

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

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

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

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

where σ is the standard deviation and, as usual, we assume that coordinates x and y are integers. To generate, say, a 3 × 3 filter mask from this function, we sample it about its center. Thus, $w_1 = h(-1, -1)$, $w_2 = h(-1, 0), \ldots, w_9 = h(1, 1)$. An $m \times n$ filter mask is generated in a similar manner. Recall that a 2-D Gaussian function has a bell shape, and that the standard deviation controls the "tightness" of the bell.

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

3.5 Smoothing Spatial Filters

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

3.5.1 Smoothing Linear Filters

The output (response) of a smoothing, linear spatial filter is simply the average of the pixels contained in the neighborhood of the filter mask. These filters sometimes are called *averaging filters*. As mentioned in the previous section, they also are referred to a *lowpass filters*.

The idea behind smoothing filters is straightforward. By replacing the value of every pixel in an image by the average of the intensity levels in the neighborhood defined by the filter mask, this process results in an image with reduced "sharp" transitions in intensities. Because random noise typically consists of sharp transitions in intensity levels, the most obvious application of smoothing is noise reduction. However, edges (which almost always are desirable features of an image) also are characterized by sharp intensity transitions, so averaging filters have the undesirable side effect that they blur edges. Another application of this type of process includes the smoothing of false contours that result from using an insufficient number of intensity levels, as discussed in Section 2.4.3. A major use of averaging filters is in the reduction of "irrelevant" detail in an image. By "irrelevant" we mean pixel regions that are small with respect to the size of the filter mask. This latter application is illustrated later in this section.

Figure 3.32 shows two 3×3 smoothing filters. Use of the first filter yields the standard average of the pixels under the mask. This can best be seen by substituting the coefficients of the mask into Eq. (3.4-4):

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

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

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

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 second mask in Fig. 3.32 is a little more interesting. This mask yields a so-called *weighted average*, terminology used to indicate that pixels are multiplied by different coefficients, thus giving more importance (weight) to some pixels at the expense of others. In the mask shown in Fig. 3.32(b) the pixel at the center of the mask is multiplied by a higher value than any other, thus giving this pixel more importance in the calculation of the average. The other pixels are inversely weighted as a function of their distance from the center of the mask. The diagonal terms are further away from the center than the orthogonal neighbors (by a factor of $\sqrt{2}$) and, thus, are weighed less than the immediate neighbors of the center pixel. The basic strategy behind weighing the center point the highest and then reducing the value of the coefficients as a function of increasing distance from the origin is simply an attempt to reduce blurring in the smoothing process. We could have chosen other weights to accomplish the same general objective. However, the sum of all the coefficients in the mask of Fig. 3.32(b) is equal to 16, an attractive feature for computer implementation because it is an integer power of 2. In practice, it is difficult in general to see differences between images smoothed by using either of the masks in Fig. 3.32, or similar arrangements, because the area spanned by these masks at any one location in an image is so small.

With reference to Eq. (3.4-1), the general implementation for filtering an $M \times N$ image with a weighted averaging filter of size $m \times n$ (m and n odd) is given by the expression

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

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

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

EXAMPLE 3.13: Image smoothing with masks of various sizes.

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

The result for $m = 5$ is somewhat similar, with a slight further increase in blurring. For $m = 9$ we see considerably more blurring, and the 20% black circle is not nearly as distinct from the background as in the previous three images, illustrating the blending effect that blurring has on objects whose intensities are close to that of its neighboring pixels. Note the significant further smoothing of the noisy rectangles. The results for $m = 15$ and 35 are extreme with respect to the sizes of the objects in the image. This type of aggresive blurring generally is used to eliminate small objects from an image. For instance, the three small squares, two of the circles, and most of the noisy rectangle areas have been blended into the background of the image in Fig. 3.33(f). Note also in this figure the pronounced black border. This is a result of padding the border of the original image with 0s (black) and then trimming off the padded area after filtering. Some of the black was blended into all filtered images, but became truly objectionable for the images smoothed with the larger filters. ■

As mentioned earlier, an important application of spatial averaging is to blur an image for the purpose of getting a gross representation of objects of interest, such that the intensity of smaller objects blends with the background and larger objects become "bloblike" and easy to detect. The size of the mask establishes the relative size of the objects that will be blended with the background. As an illustration, consider Fig. 3.34(a), which is an image from the Hubble telescope in orbit around the Earth. Figure 3.34(b) shows the result of applying a 15×15 averaging mask to this image. We see that a number of objects have either blended with the background or their intensity has diminished considerably. It is typical to follow an operation like this with thresholding to eliminate objects based on their intensity. The result of using the thresholding function of Fig. 3.2(b) with a threshold value equal to 25% of the highest intensity in the blurred image is shown in Fig. 3.34(c). Comparing this result with the original image, we see that it is a reasonable representation of what we would consider to be the largest, brightest objects in that image.

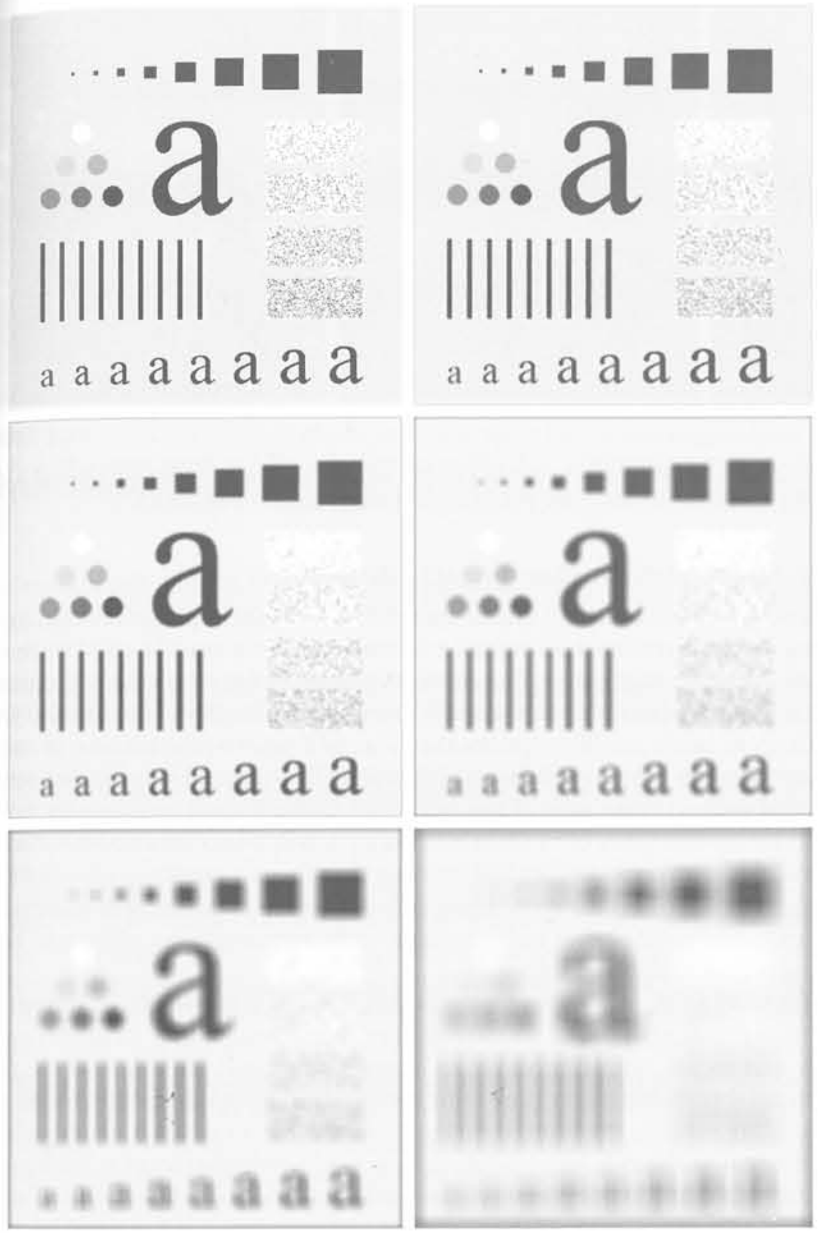
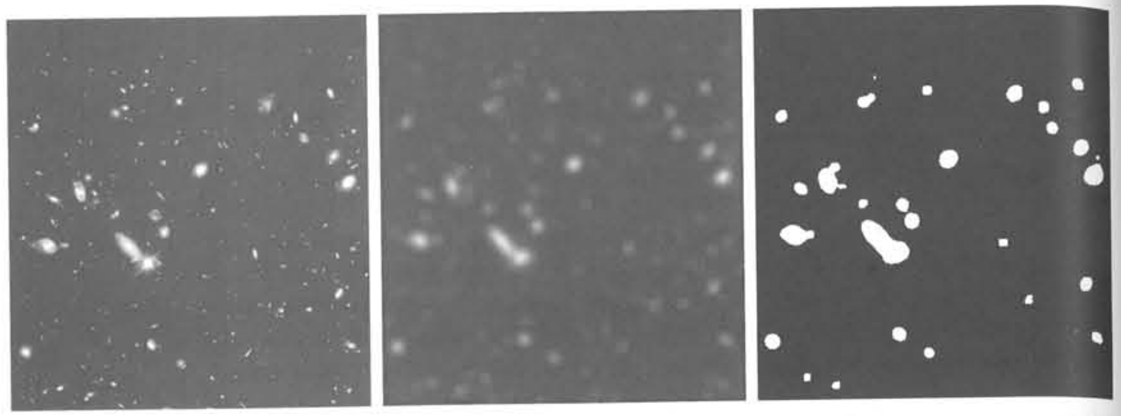


FIGURE 3.33 (a) Original image, of size 500×500 pixels. (b)–(f) Results of smoothing with square averaging filter masks of sizes $m = 3, 5, 9, 15$, and 35, respectively. The black squares at the top are of sizes 3, 5, 9, 15, 25, 35, 45, and 55 pixels, respectively; their borders are 25 pixels apart. The letters at the bottom range in size from 10 to 24 points, in increments of 2 points; the large letter at the top is 60 points. The vertical bars are 5 pixels wide and 100 pixels high; their separation is 20 pixels. The diameter of the circles is 25 pixels, and their borders are 15 pixels apart; their intensity levels range from 0% to 100% black in increments of 20%. The background of the image is 10% black. The noisy rectangles are of size 50×120 pixels.

a b
c d
e f



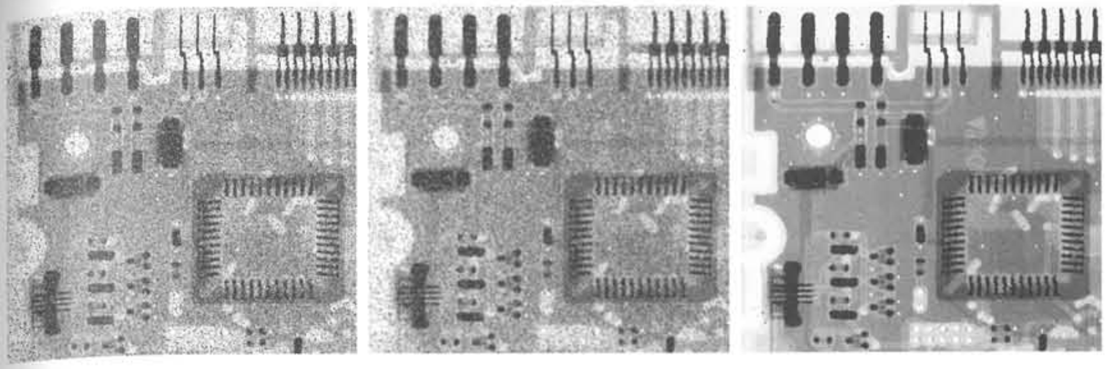
a b c

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

3.5.2 Order-Statistic (Nonlinear) Filters

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

The median, ξ, of a set of values is such that half the values in the set are less than or equal to ξ, and half are greater than or equal to ξ. In order to perform median filtering at a point in an image, we first sort the values of the pixel in the neighborhood, determine their median, and assign that value to the corresponding pixel in the filtered image. For example, in a 3×3 neighborhood the median is the 5th largest value, in a 5×5 neighborhood it is the 13th largest value, and so on. When several values in a neighborhood are the same, all equal values are grouped. For example, suppose that a 3×3 neighborhood has values (10, 20, 20, 20, 15, 20, 20, 25, 100). These values are sorted as (10, 15, 20, 20, 20, 20, 20, 25, 100), which results in a median of 20. Thus, the principal function of median filters is to force points with distinct intensity levels to be more like their neighbors. In fact, isolated clusters of pixels that are light or dark with respect to their neighbors, and whose area is less than $m^2/2$ (one-half the filter area), are eliminated by an $m \times m$ median filter. In this case "eliminated" means forced to the median intensity of the neighbors. Larger clusters are affected considerably less.



a b c

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

Although the median filter is by far the most useful order-statistic filter in image processing, it is by no means the only one. The median represents the 50th percentile of a ranked set of numbers, but recall from basic statistics that ranking lends itself to many other possibilities. For example, using the 100th percentile results in the so-called *max filter*, which is useful for finding the brightest points in an image. The response of a 3×3 max filter is given by $R = \max\{z_k | k = 1, 2, \ldots, 9\}$. The 0th percentile filter is the *min filter*, used for the opposite purpose. Median, max, min, and several other nonlinear filters are considered in more detail in Section 5.3.

See Section 10.3.5 regarding percentiles.

EXAMPLE 3.14: Use of median filtering for noise reduction.

■ Figure 3.35(a) shows an X-ray image of a circuit board heavily corrupted by salt-and-pepper noise. To illustrate the point about the superiority of median filtering over average filtering in situations such as this, we show in Fig. 3.35(b) the result of processing the noisy image with a 3×3 neighborhood averaging mask, and in Fig. 3.35(c) the result of using a 3×3 median filter. The averaging filter blurred the image and its noise reduction performance was poor. The superiority in all respects of median over average filtering in this case is quite evident. In general, median filtering is much better suited than averaging for the removal of salt-and-pepper noise. ■

3.6 Sharpening Spatial Filters

The principal objective of sharpening is to highlight transitions in intensity. Uses of image sharpening vary and include applications ranging from electronic printing and medical imaging to industrial inspection and autonomous guidance in military systems. In the last section, we saw that image blurring could be accomplished in the spatial domain by pixel averaging in a neighborhood. Because averaging is analogous to integration, it is logical to conclude that sharpening can be accomplished by spatial differentiation. This, in fact, is the case,