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

3.5 Basics of Spatial Filtering

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

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

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

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

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

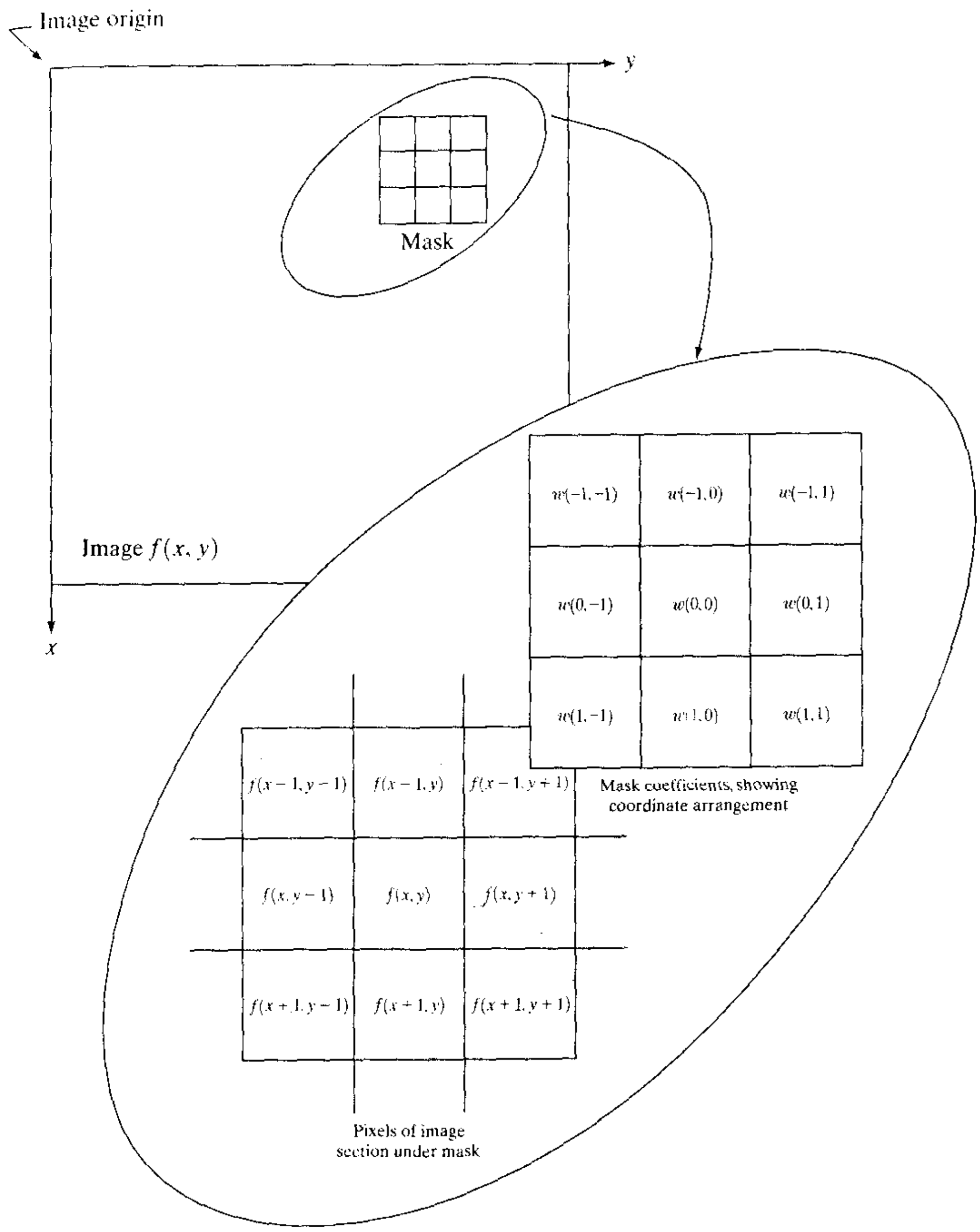


FIGURE 3.32 The mechanics of spatial filtering. The magnified drawing shows a 3×3 mask and the image section directly under it; the image section is shown displaced out from under the mask for ease of readability.

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

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

where, from the previous paragraph, $a = (m - 1)/2$ and $b = (n - 1)/2$. To generate a complete filtered image this equation must be applied for $x = 0, 1, 2, \ldots, M - 1$ and $y = 0, 1, 2, \ldots, N - 1$. In this way, we are assured that the mask processes all pixels in the image. It is easily verified when $m = n = 3$ that this expression reduces to the example given in the previous paragraph.

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

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

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

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

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

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

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

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

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

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

Smoothing Spatial Filters

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

Smoothing Linear Filters

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

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

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

a b

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

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

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

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

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

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

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

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

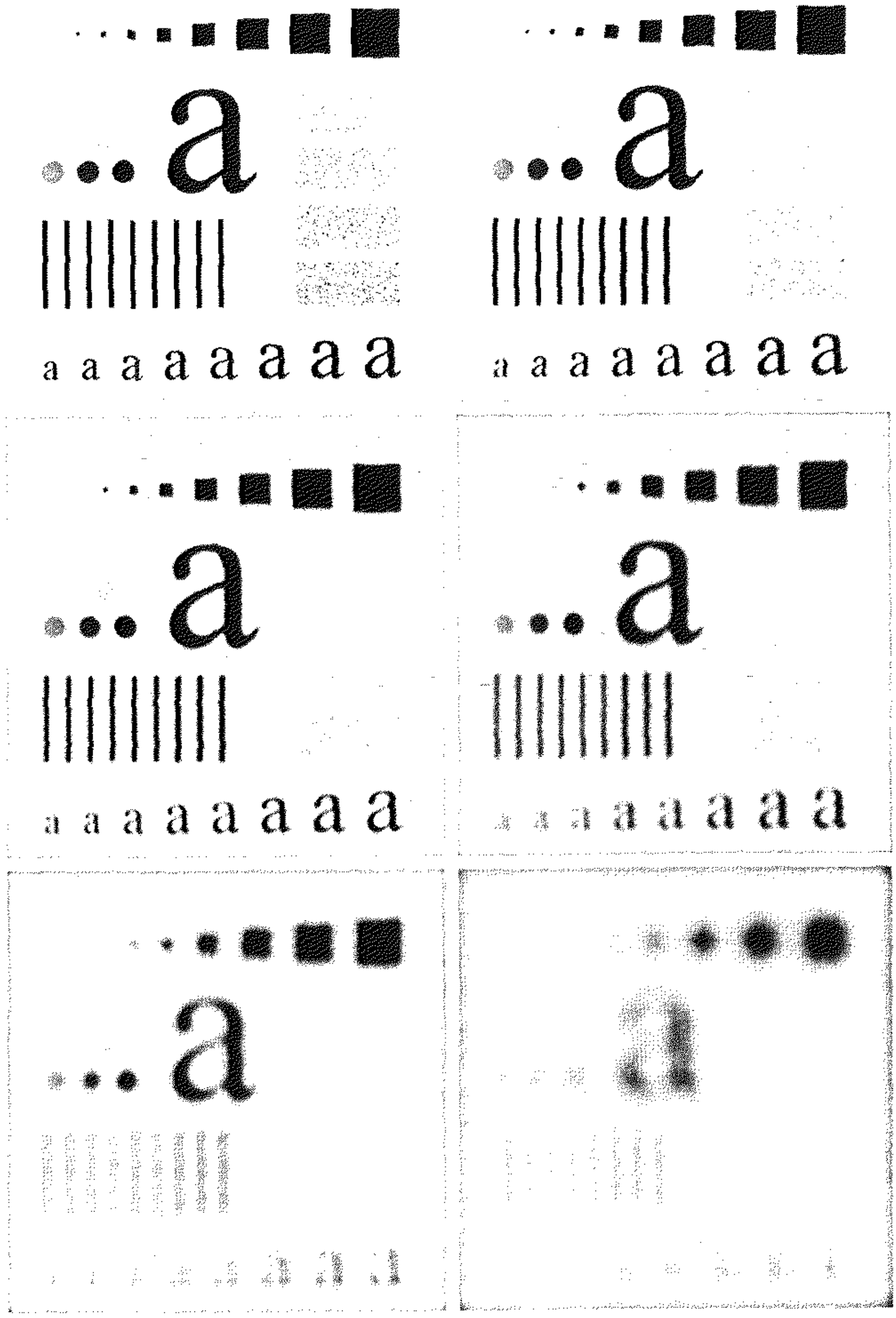
The parameters in this equation are as defined in Eq. (3.5-1). As before, it is understood that the complete filtered image is obtained by applying Eq. (3.6-1) for $x = 0, 1, 2, \ldots, M - 1$ and $y = 0, 1, 2, \ldots, N - 1$. The denominator in Eq. (3.6-1) is simply the sum of the mask coefficients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

EXAMPLE 3.9: Image smoothing with masks of various sizes.

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

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

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



a b
c d
e f

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