THE MECHANICS OF LINEAR SPATIAL FILTERING

A linear spatial filter performs a sum-of-products operation between an image f and a *filter kernel*, w. The kernel is an array whose size defines the neighborhood of operation, and whose coefficients determine the nature of the filter. Other terms used to refer to a spatial filter kernel are *mask*, *template*, and *window*. We use the term *filter kernel* or simply *kernel*.

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

$$\begin{aligned} g(x,y) = {} & w(-1,-1)f(x-1,y-1) + w(-1,0)f(x-1,y) + \ldots \\ & + w(0,0)f(x,y) + \ldots + w(1,1)f(x+1,y+1) \end{aligned} \tag{3-30}$$

As coordinates x and y are varied, the center of the kernel moves from pixel to pixel, generating the filtered image, g, in the process.[†]

Observe that the center coefficient of the kernel, $w(0,0)$, aligns with the pixel at location (x, y). For a kernel of size $m \times n$, we assume that $m = 2a + 1$ and $n = 2b + 1$, where a and b are nonnegative integers. This means that our focus is on kernels of odd size in both coordinate directions. In general, linear spatial filtering of an image of size $M \times N$ with a kernel of size $m \times n$ is given by the expression

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

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t) \tag{3-31}$$

where x and y are varied so that the center (origin) of the kernel visits every pixel in f once. For a fixed value of (x, y), Eq. (3-31) implements the *sum of products* of the form shown in Eq. (3-30), but for a kernel of arbitrary odd size. As you will learn in the following section, this equation is a central tool in linear filtering.

SPATIAL CORRELATION AND CONVOLUTION

Spatial correlation is illustrated graphically in Fig. 3.28, and it is described mathematically by Eq. (3-31). Correlation consists of moving the center of a kernel over an image, and computing the sum of products at each location. The mechanics of *spatial convolution* are the same, except that the correlation kernel is rotated by 180°. Thus, when the values of a kernel are symmetric about its center, correlation and convolution yield the same result. The reason for rotating the kernel will become clear in the following discussion. The best way to explain the differences between the two concepts is by example.

We begin with a 1-D illustration, in which case Eq. (3-31) becomes

$$g(x) = \sum_{s=-a}^{a} w(s) f(x+s) \tag{3-32}$$

[†] A 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 is being performed.

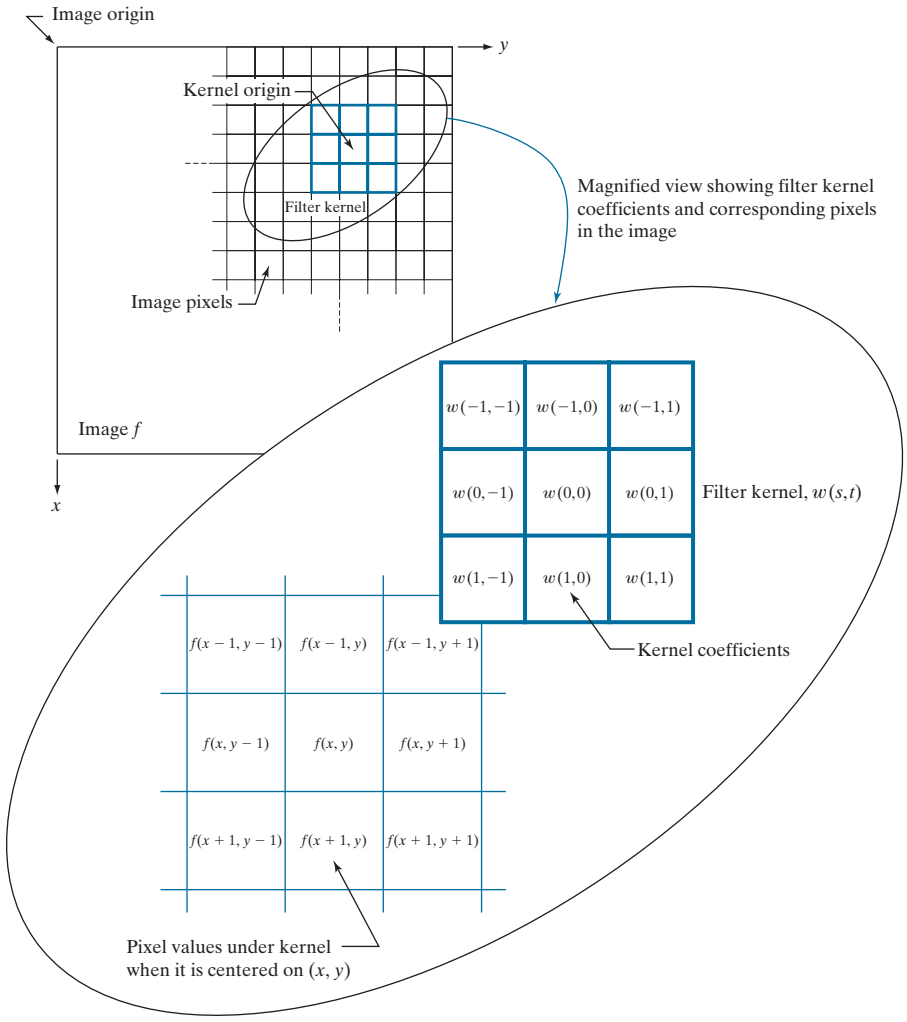

FIGURE 3.28 The mechanics of linear spatial filtering using a 3×3 kernel. The pixels are shown as squares to simplify the graphics. Note that the origin of the image is at the top left, but the origin of the kernel is at its center. Placing the origin at the center of spatially symmetric kernels simplifies writing expressions for linear filtering.

Figure 3.29(a) shows a 1-D function, f, and a kernel, w. The kernel is of size 1×5, so $a = 2$ and $b = 0$ in this case. Figure 3.29(b) shows the starting position used to perform correlation, in which w is positioned so that its center coefficient is coincident with the origin of f.

The first thing we notice is that part of w lies outside f, so the summation is undefined in that area. A solution to this problem is to *pad* function f with enough 0's on either side. In general, if the kernel is of size $1 \times m$, we need $(m-1)/2$ zeros on either side of f in order to handle the beginning and ending configurations of w with respect to f. Figure 3.29(c) shows a properly padded function. In this starting configuration, all coefficients of the kernel overlap valid values.

Zero padding is not the only padding option, as we will discuss in detail later in this chapter.

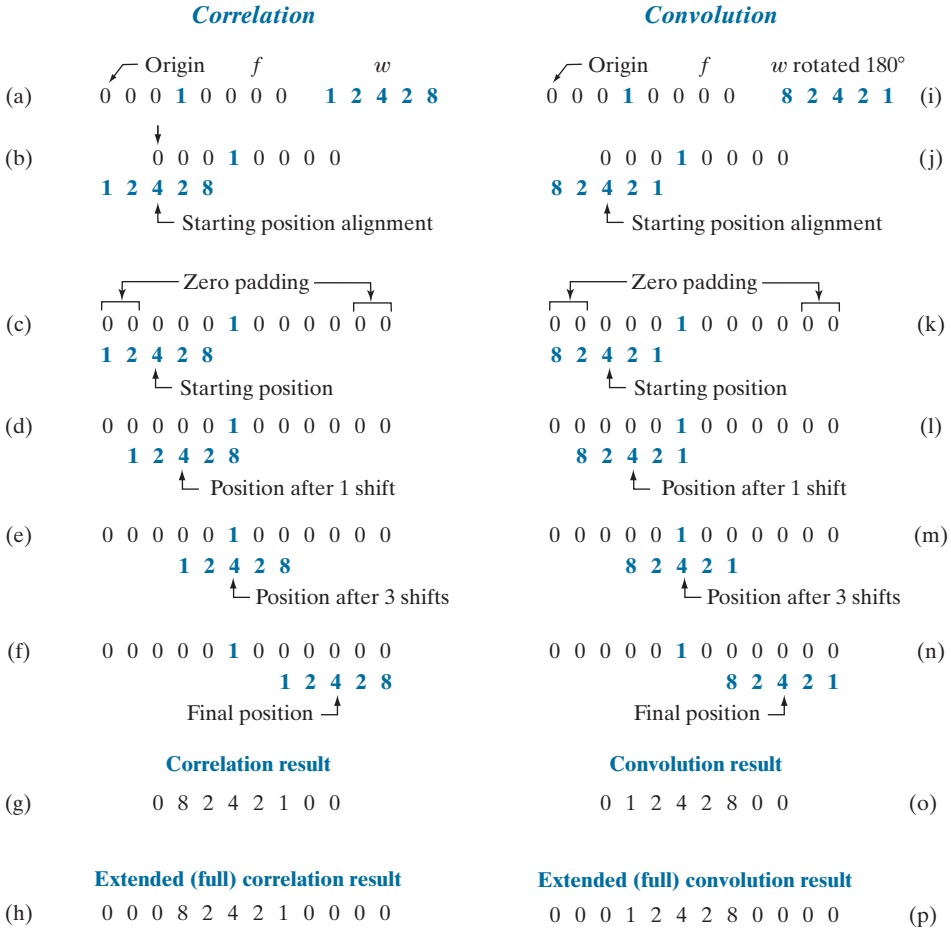

FIGURE 3.29 Illustration of 1-D correlation and convolution of a kernel, w, with a function f consisting of a discrete unit impulse. Note that correlation and convolution are functions of the variable x, which acts to *displace* one function with respect to the other. For the extended correlation and convolution results, the starting configuration places the rightmost element of the kernel to be coincident with the origin of f. Additional padding must be used.

The first correlation value is the sum of products in this initial position, computed using Eq. (3-32) with $x = 0$:

$$g(0) = \sum_{s=-2}^{2} w(s)f(s+0) = 0$$

This value is in the leftmost location of the correlation result in Fig. 3.29(g).

To obtain the second value of correlation, we shift the relative positions of w and f one pixel location to the right [i.e., we let $x = 1$ in Eq. (3-32)] and compute the sum of products again. The result is $g(1) = 8$, as shown in the leftmost, nonzero location in Fig. 3.29(g). When $x = 2$, we obtain $g(2) = 2$. When $x = 3$, we get $g(3) = 4$ [see Fig. 3.29(e)]. Proceeding in this manner by varying x one shift at a time, we "build" the correlation result in Fig. 3.29(g). Note that it took 8 values of x (i.e., $x = 0, 1, 2, \ldots, 7$) to fully shift w past f so the *center* coefficient in w visited *every* pixel in f. Sometimes, it is useful to have every element of w visit every pixel in f. For this, we have to start

with the rightmost element of w coincident with the origin of f, and end with the leftmost element of w being coincident the last element of f (additional padding would be required). Figure Fig. 3.29(h) shows the result of this *extended*, or *full*, correlation. As Fig. 3.29(g) shows, we can obtain the "standard" correlation by cropping the full correlation in Fig. 3.29(h).

There are two important points to note from the preceding discussion. First, correlation is a function of *displacement* of the filter kernel relative to the image. In other words, the first value of correlation corresponds to zero displacement of the kernel, the second corresponds to one unit displacement, and so on.[†] The second thing to notice is that correlating a kernel w with a function that contains all 0's and a single 1 yields a *copy* of w, but *rotated* by 180°. A function that contains a single 1 with the rest being 0's is called a *discrete unit impulse*. Correlating a kernel with a discrete unit impulse yields a *rotated* version of the kernel at the location of the impulse.

Rotating a 1-D kernel by 180° is equivalent to flipping the kernel about its axis.

The right side of Fig. 3.29 shows the sequence of steps for performing convolution (we will give the equation for convolution shortly). The only difference here is that the kernel is *pre-rotated* by 180° prior to performing the shifting/sum of products operations. As the convolution in Fig. 3.29(o) shows, the result of pre-rotating the kernel is that now we have an *exact* copy of the kernel at the location of the unit impulse. In fact, a foundation of linear system theory is that convolving a function with an impulse yields a copy of the function at the location of the impulse. We will use this property extensively in Chapter 4.

The 1-D concepts just discussed extend easily to images, as Fig. 3.30 shows. For a kernel of size $m \times n$, we pad the image with a minimum of $(m-1)/2$ rows of 0's at the top and bottom and $(n-1)/2$ columns of 0's on the left and right. In this case, m and n are equal to 3, so we pad f with one row of 0's above and below and one column of 0's to the left and right, as Fig. 3.30(b) shows. Figure 3.30(c) shows the initial position of the kernel for performing correlation, and Fig. 3.30(d) shows the final result after the center of w visits every pixel in f, computing a sum of products at each location. As before, the result is a copy of the kernel, rotated by 180°. We will discuss the extended correlation result shortly.

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

For convolution, we pre-rotate the kernel 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 to the location of the impulse. As noted earlier, correlation and convolution yield the same result if the kernel values are symmetric about the center.

The concept of an impulse is fundamental in linear system theory, and is used in numerous places throughout the book. A *discrete impulse of strength* (*amplitude*) A located at coordinates (x_0, y_0) is defined as

$$\delta(x - x_0, y - y_0) = \begin{cases} A & \text{if } x = x_0 \text{ and } y = y_0 \\ 0 & \text{otherwise} \end{cases} \tag{3-33}$$

[†] In reality, we are shifting f to the left of w every time we increment x in Eq. (3-32). However, it is more intuitive to think of the smaller kernel moving right over the larger array f. The motion of the two is relative, so either way of looking at the motion is acceptable. The reason we increment f and not w is that indexing the equations for correlation and convolution is much easier (and clearer) this way, especially when working with 2-D arrays.

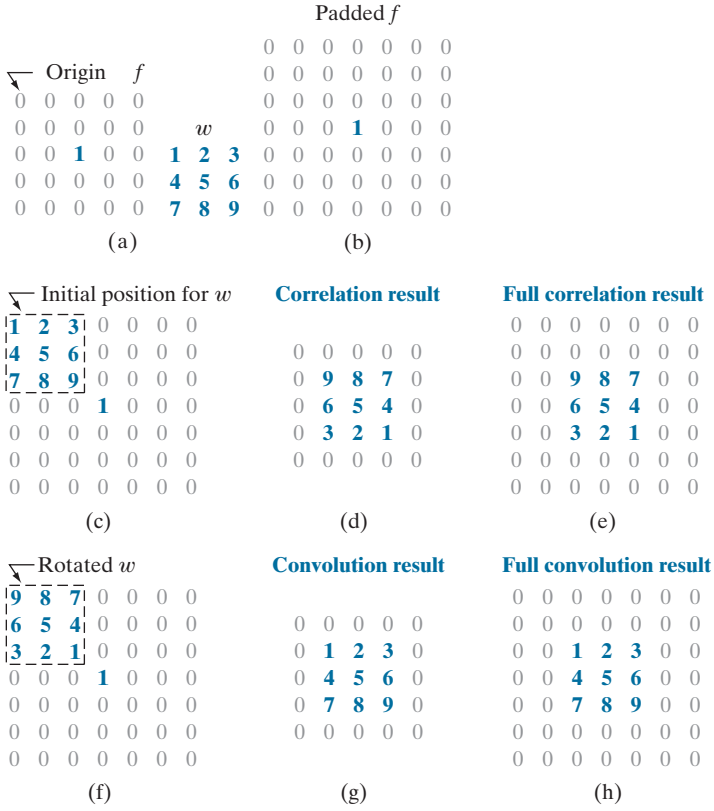

FIGURE 3.30 Correlation (middle row) and convolution (last row) of a 2-D kernel with an image consisting of a discrete unit impulse. The 0's are shown in gray to simplify visual analysis. Note that correlation and convolution are functions of x and y. As these variable change, they *displace* one function with respect to the other. See the discussion of Eqs. (3-36) and (3-37) regarding full correlation and convolution.

Recall that $A = 1$ for a unit impulse.

For example, the unit impulse in Fig. 3.29(a) is given by $\delta(x-3)$ in the 1-D version of the preceding equation. Similarly, the impulse in Fig. 3.30(a) is given by $\delta(x-2, y-2)$ [remember, the origin is at $(0,0)$].

Summarizing the preceding discussion in equation form, the correlation of a kernel w of size $m \times n$ with an image $f(x,y)$, denoted as $(w \star f)(x,y)$, is given by Eq. (3-31), which we repeat here for convenience:

$$(w \star f)(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t) \tag{3-34}$$

Because our kernels do not depend on (x,y), we will sometimes make this fact explicit by writing the left side of the preceding equation as $w \star f(x,y)$. Equation (3-34) is evaluated for all values of the displacement variables x and y so that the center point of w visits every pixel in f,[†] where we assume that f has been padded appropriately.

[†] As we mentioned earlier, the *minimum* number of required padding elements for a 2-D correlation is $(m-1)/2$ rows above and below f, and $(n-1)/2$ columns on the left and right. With this padding, and assuming that f is of size $M \times N$, the values of x and y required to obtain a complete correlation are $x = 0, 1, 2, \ldots, M-1$ and $y = 0, 1, 2, \ldots, N-1$. This assumes that the starting configuration is such that the *center* of the kernel coincides with the *origin* of the image, which we have defined to be at the top, left (see Fig. 2.19).