# 7 Wavelets and Multiresolution Processing

All this time, the guard was looking at her, first through a telescope, then through a microscope, and then through an opera glass.

*Lewis Carrol, Through the Looking Glass*

## *Preview*

Although the Fourier transform has been the mainstay of transform-based image processing since the late 1950s, a more recent transformation, called the *wavelet transform*, is now making it even easier to compress, transmit, and analyze many images. Unlike the Fourier transform, whose basis functions are sinusoids, wavelet transforms are based on small waves, called *wavelets*, of varying frequency *and limited duration*. This allows them to provide the equivalent of a musical score for an image, revealing not only what notes (or frequencies) to play but also when to play them. Fourier transforms, on the other hand, provide only the notes or frequency information; temporal information is lost in the transformation process.

In 1987, wavelets were first shown to be the foundation of a powerful new approach to signal processing and analysis called *multiresolution* theory (Mallat [1987]). Multiresolution theory incorporates and unifies techniques from a variety of disciplines, including subband coding from signal processing, quadrature mirror filtering from digital speech recognition, and pyramidal image processing. As its name implies, multiresolution theory is concerned with the representation and analysis of signals (or images) at more than one resolution. The appeal of such an approach is obvious—features that might go undetected at one resolution may be easy to detect at another. Although the imaging community's interest in multiresolution analysis was limited until the late 1980s, it is now difficult to keep up with the number of papers, theses, and books devoted to the subject.

In this chapter, we examine wavelet-based transformations from a multiresolution point of view. Although such transformations can be presented in other ways, this approach simplifies both their mathematical and physical interpretations. We begin with an overview of imaging techniques that influenced the formulation of multiresolution theory. Our objective is to introduce the theory's fundamental concepts within the context of image processing and simultaneously provide a brief historical perspective of the method and its application. The bulk of the chapter is focused on the development and use of the discrete wavelet transform. To demonstrate the usefulness of the transform, examples ranging from image coding to noise removal and edge detection are provided. In the next chapter, wavelets will be used for image compression, an application in which they have received considerable attention.

## 7.1 Background

When we look at images, generally we see connected regions of similar texture and intensity levels that combine to form objects. If the objects are small in size or low in contrast, we normally examine them at high resolutions; if they are large in size or high in contrast, a coarse view is all that is required. If both small and large objects—or low- and high-contrast objects—are present simultaneously, it can be advantageous to study them at several resolutions. This, of course, is the fundamental motivation for multiresolution processing.

From a mathematical viewpoint, images are two-dimensional arrays of intensity values with locally varying statistics that result from different combinations of abrupt features like edges and contrasting homogeneous regions. As illustrated in Fig. 7.1—an image that will be examined repeatedly in the remainder of the

Local histograms are histograms of the pixels in a neighborhood (see Section 3.3.3).

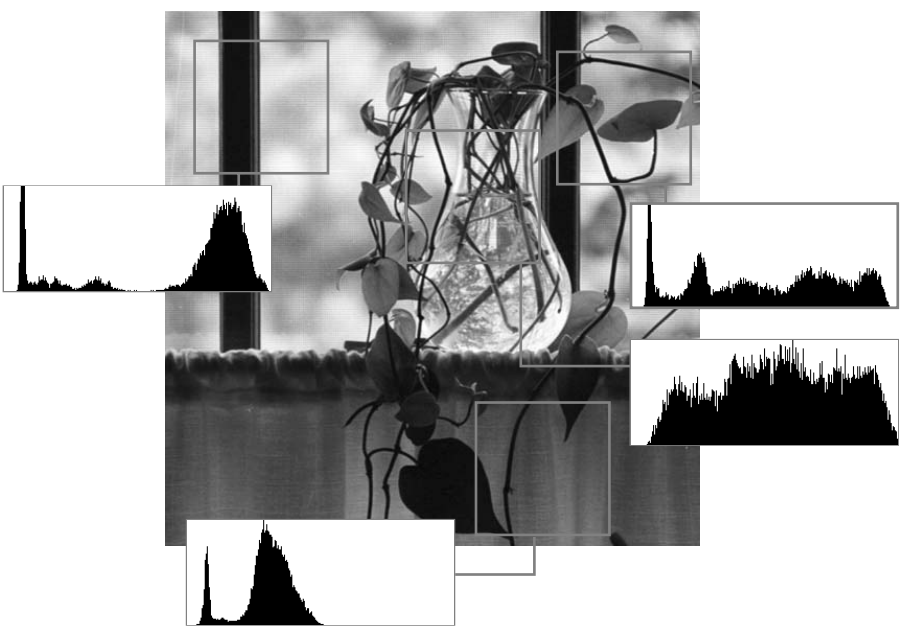


**FIGURE 7.1** An image and its local histogram variations.

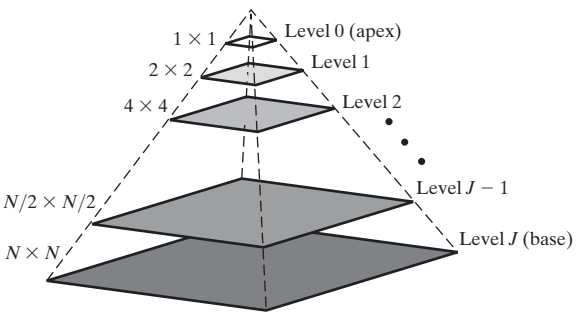
section—local histograms can vary significantly from one part of an image to another, making statistical modeling over the span of an entire image a difficult, or impossible task.

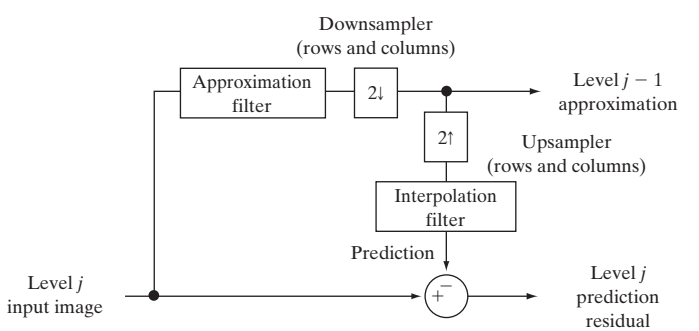
### 7.1.1 Image Pyramids

A powerful, yet conceptually simple structure for representing images at more than one resolution is the *image pyramid* (Burt and Adelson [1983]). Originally devised for machine vision and image compression applications, an image pyramid is a collection of decreasing resolution images arranged in the shape of a pyramid. As can be seen in Fig. 7.2(a), the base of the pyramid contains a high-resolution representation of the image being processed; the apex contains a low-resolution approximation. As you move up the pyramid, both size and resolution decrease. Base level $J$ is of size $2^J \times 2^J$ or $N \times N$, where $J = \log_2 N$, apex level 0 is of size $1 \times 1$, and general level $j$ is of size $2^j \times 2^j$, where $0 \leq j \leq J$. Although the pyramid shown in Fig. 7.2(a) is composed of $J + 1$ resolution levels from $2^J \times 2^J$ to $2^0 \times 2^0$, most image pyramids are truncated to $P + 1$ levels, where $1 \leq P \leq J$ and $j = J - P, \ldots, J - 2, J - 1, J$. That is, we normally limit ourselves to $P$ reduced resolution approximations of the original image; a $1 \times 1$ (i.e., single pixel) approximation of a $512 \times 512$ image, for example, is of little value. The total number of pixels in a $P + 1$ level pyramid for $P > 0$ is

$$N^2\left(1 + \frac{1}{(4)^1} + \frac{1}{(4)^2} + \cdots + \frac{1}{(4)^P}\right) \leq \frac{4}{3}N^2$$

Figure 7.2(b) shows a simple system for constructing two intimately related image pyramids. The *Level $j - 1$ approximation* output provides the images




a
b

**FIGURE 7.2** (a) An image pyramid. (b) A simple system for creating approximation and prediction residual pyramids.

needed to build an *approximation pyramid* (as described in the preceding paragraph), while the *Level j prediction residual* output is used to build a complementary *prediction residual pyramid*. Unlike approximation pyramids, prediction residual pyramids contain only one reduced-resolution approximation of the input image (at the top of the pyramid, level $J - P$). All other levels contain prediction residuals, where the level $j$ *prediction residual* (for $J - P + 1 \leq j \leq J$) is defined as the difference between the level $j$ approximation (the input to the block diagram) and an estimate of the level $j$ approximation based on the level $j - 1$ approximation (the approximation output in the block diagram).

In general, a prediction residual can be defined as the difference between an image and a predicted version of the image. As will be seen in Section 8.2.9, prediction residuals can often be coded more efficiently than 2-D intensity arrays.

As Fig. 7.2(b) suggests, both approximation and prediction residual pyramids are computed in an iterative fashion. Before the first iteration, the image to be represented in pyramidal form is placed in level $J$ of the approximation pyramid. The following three-step procedure is then executed $P$ times—for $j = J, J - 1, \ldots,$ and $J - P + 1$ (in that order):

**Step 1.** Compute a reduced-resolution approximation of the *Level j input image* [the input on the left side of the block diagram in Fig. 7.2(b)]. This is done by filtering and downsampling the filtered result by a factor of 2. Both of these operations are described in the next paragraph. Place the resulting approximation at level $j - 1$ of the approximation pyramid.

**Step 2.** Create an estimate of the *Level j input image* from the reduced-resolution approximation generated in step 1. This is done by upsampling and filtering (see the next paragraph) the generated approximation. The resulting prediction image will have the same dimensions as the *Level j input image*.

**Step 3.** Compute the difference between the prediction image of step 2 and the input to step 1. Place this result in level $j$ of the prediction residual pyramid.

At the conclusion of $P$ iterations (i.e., following the iteration in which $j = J - P + 1$), the level $J - P$ approximation output is placed in the prediction residual pyramid at level $J - P$. If a prediction residual pyramid is not needed, this operation—along with steps 2 and 3 and the upsampler, interpolation filter, and summer of Fig. 7.2(b)—can be omitted.

A variety of approximation and interpolation filters can be incorporated into the system of Fig. 7.2(b). Typically, the filtering is performed in the spatial domain (see Section 3.4). Useful approximation filtering techniques include neighborhood averaging (see Section 3.5.1.), which produces *mean pyramids*; lowpass Gaussian filtering (see Sections 4.7.4 and 4.8.3), which produces *Gaussian pyramids*; and no filtering, which results in *subsampling pyramids*. Any of the interpolation methods described in Section 2.4.4, including nearest neighbor, bilinear, and bicubic, can be incorporated into the interpolation filter. Finally, we note that the upsampling and downsampling blocks of Fig. 7.2(b) are used to double and halve the spatial dimensions of the approximation and prediction images that are computed. Given an integer variable $n$ and 1-D sequence of samples $f(n)$, *upsampled* sequence $f_{2\uparrow}(n)$ is defined as

$$f_{2\uparrow}(n) = \begin{cases} f(n/2) & \text{if } n \text{ is even} \\ 0 & \text{otherwise} \end{cases} \tag{7.1-1}$$

In this chapter, we will be working with both continuous and discrete functions and variables. With the notable exception of 2-D image $f(x, y)$ and unless otherwise noted, $x, y, z, \ldots$ are continuous variables; $i, j, k, l, m, n, \ldots$ are discrete variables.

where, as is indicated by the subscript, the upsampling is by a factor of 2. The complementary operation of *downsampling* by 2 is defined as

$$f_{2\downarrow}(n) = f(2n) \tag{7.1-2}$$

Upsampling can be thought of as inserting a 0 after every sample in a sequence; downsampling can be viewed as discarding every other sample. The upsampling and downsampling blocks in Fig. 7.2(b), which are labeled $2\uparrow$ and $2\downarrow$, respectively, are annotated to indicate that both the rows and columns of the 2-D inputs on which they operate are to be up- and downsampled. Like the separable 2-D DFT in Section 4.11.1, 2-D upsampling and downsampling can be performed by successive passes of the 1-D operations defined in Eqs. (7.1-1) and (7.1-2).

**EXAMPLE 7.1:** Approximation and prediction residual pyramids.

■ Figure 7.3 shows both an approximation pyramid and a prediction residual pyramid for the vase of Fig. 7.1. A lowpass Gaussian smoothing filter (see Section 4.7.4) was used to produce the four-level approximation pyramid in Fig. 7.3(a). As you can see, the resulting pyramid contains the original $512 \times 512$ resolution image (at its base) and three low-resolution approximations (of resolution $256 \times 256$, $128 \times 128$, and $64 \times 64$). Thus, $P$ is 3 and levels 9, 8, 7, and 6 out of a possible $\log_2(512) + 1$ or 10 levels are present. Note the reduction in detail that accompanies the lower resolutions of the pyramid. The level 6 (i.e., $64 \times 64$) approximation image is suitable for locating the window stiles (i.e., the window pane framing), for example, but not for finding the stems of the plant. In general, the lower-resolution levels of a pyramid can be used for the analysis of large structures or overall image context; the high-resolution images are appropriate for analyzing individual object characteristics. Such a coarse-to-fine analysis strategy is particularly useful in pattern recognition.

A bilinear interpolation filter was used to produce the prediction residual pyramid in Fig. 7.3(b). In the absence of quantization error, the resulting prediction residual pyramid can be used to generate the complementary approximation pyramid in Fig. 7.3(a), including the original image, without error. To do so, we begin with the level 6 $64 \times 64$ approximation image (the only approximation image in the prediction residual pyramid), predict the level 7 $128 \times 128$ resolution approximation (by upsampling and filtering), and add the level 7 prediction residual. This process is repeated using successively computed approximation images until the original $512 \times 512$ image is generated. Note that the prediction residual histogram in Fig. 7.3(b) is highly peaked around zero; the approximation histogram in Fig. 7.3(a) is not. Unlike approximation images, prediction residual images can be highly compressed by assigning fewer bits to the more probable values (see the variable-length codes of Section 8.2.1). Finally, we note that the prediction residuals in Fig. 7.3(b) are scaled to make small prediction errors more visible; the prediction residual histogram, however, is based on the original residual values, with level 128 representing zero error. ■

a
b

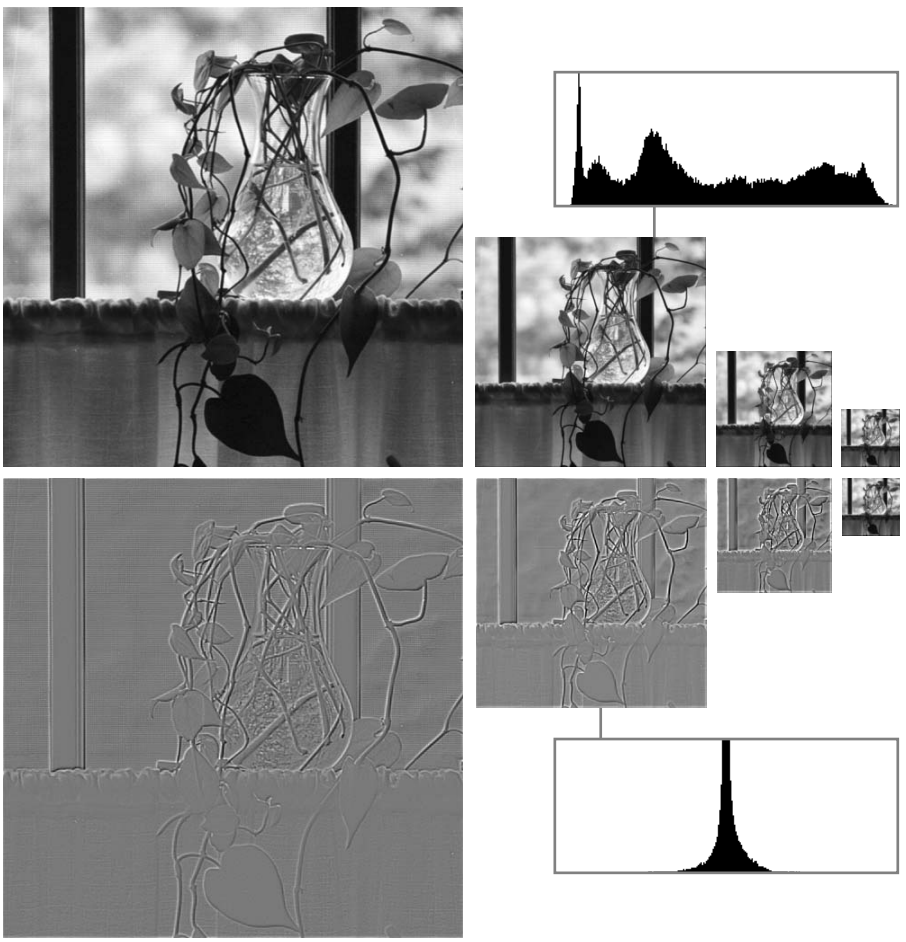


**FIGURE 7.3**
Two image pyramids and their histograms: (a) an approximation pyramid; (b) a prediction residual pyramid.

The approximation pyramid in (a) is called a Gaussian pyramid because a Gaussian filter was used to construct it. The prediction residual pyramid in (b) is often called a Laplacian pyramid; note the similarity in appearance with the Laplacian filtered images in Chapter 3.

### 7.1.2 Subband Coding

Another important imaging technique with ties to multiresolution analysis is *subband coding*. In subband coding, an image is decomposed into a set of bandlimited components, called subbands. The decomposition is performed so that the subbands can be reassembled to reconstruct the original image without error. Because the decomposition and reconstruction are performed by means of digital filters, we begin our discussion with a brief introduction to *digital signal processing* (DSP) and *digital signal filtering*.

The term "delay" implies a time-based input sequence and reflects the fact that in digital signal filtering, the input is usually a sampled analog signal.

Consider the simple *digital filter* in Fig. 7.4(a) and note that it is constructed from three basic components—*unit delays*, *multipliers*, and *adders*. Along the top of the filter, unit delays are connected in series to create $K-1$ delayed (i.e., right shifted) versions of the input sequence $f(n)$. Delayed sequence $f(n-2)$, for example, is

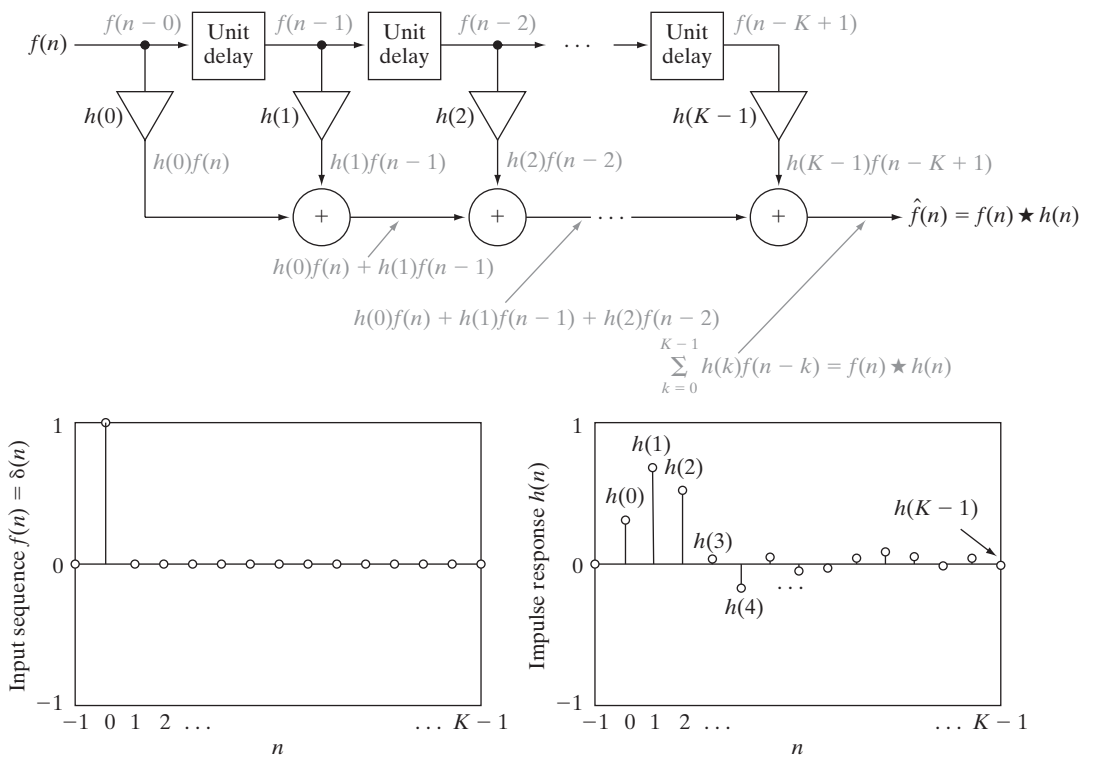
$$f(n-2) = \begin{cases} \vdots & \\ f(0) & \text{for } n = 2 \\ f(1) & \text{for } n = 2 + 1 = 3 \\ \vdots & \end{cases}$$

As the grayed annotations in Fig. 7.4(a) indicate, input sequence $f(n) = f(n-0)$ and the $K-1$ delayed sequences at the outputs of the unit delays, denoted $f(n-1), f(n-2), \ldots, f(n-K+1)$, are multiplied by constants $h(0), h(1), \ldots, h(K-1)$, respectively, and summed to produce the filtered output sequence

$$\begin{aligned} \hat{f}(n) &= \sum_{k=-\infty}^{\infty} h(k) f(n-k) \\ &= f(n) \star h(n) \end{aligned} \tag{7.1-3}$$

where ★ denotes convolution. Note that—except for a change in variables—Eq. (7.1-3) is equivalent to the discrete convolution defined in Eq. (4.4-10) of Chapter 4. The $K$ multiplication constants in Fig. 7.4(a) and Eq. (7.1-3) are

If the coefficients of the filter in Fig. 7.4(a) are indexed using values of $n$ between 0 and $K-1$ (as we have done), the limits on the sum in Eq. (7.1-3) can be reduced to 0 to $K-1$ [like Eq. (4.4-10)].




a
b c

**FIGURE 7.4** (a) A digital filter; (b) a unit discrete impulse sequence; and (c) the impulse response of the filter.

called *filter coefficients*. Each coefficient defines a *filter tap*, which can be thought of as the components needed to compute one term of the sum in Eq. (7.1-3), and the filter is said to be of *order K*.

If the input to the filter of Fig. 7.4(a) is the unit discrete impulse of Fig. 7.4(b) and Section 4.2.3, Eq. (7.1-3) becomes

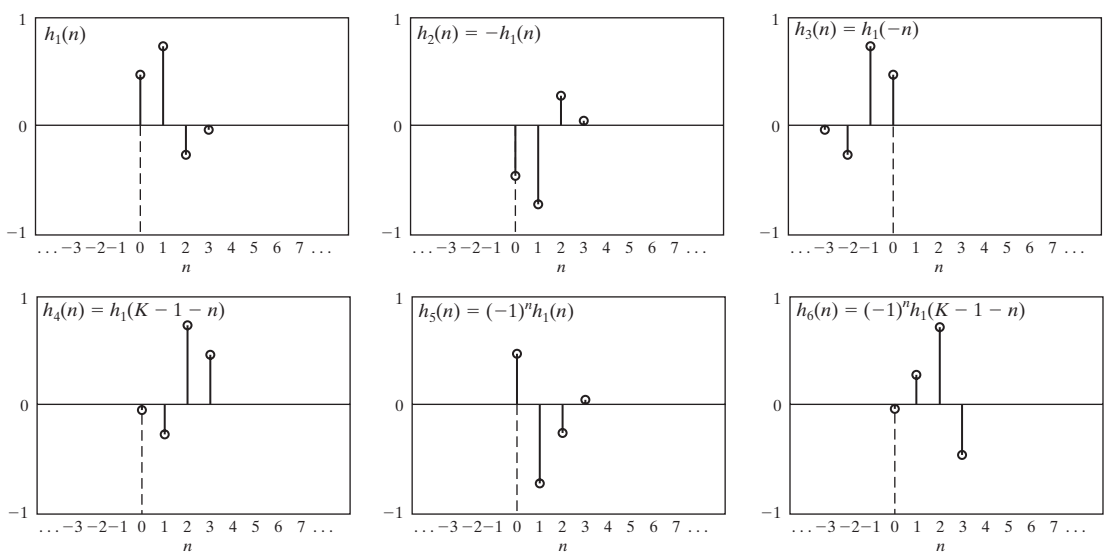
$$\begin{aligned}\hat{f}(n) &= \sum_{k=-\infty}^{\infty} h(k)\delta(n-k) \\ &= h(n)\end{aligned} \tag{7.1-4}$$

That is, by substituting $\delta(n)$ for input $f(n)$ in Eq. (7.1-3) and making use of the sifting property of the unit discrete impulse as defined in Eq. (4.2-13), we find that the *impulse response* of the filter in Fig. 7.4(a) is the $K$-element sequence of filter coefficients that define the filter. Physically, the unit impulse is shifted from left to right across the top of the filter (from one unit delay to the next), producing an output that assumes the value of the coefficient at the location of the delayed impulse. Because there are $K$ coefficients, the impulse response is of length $K$ and the filter is called a *finite impulse response* (FIR) filter.

In the remainder of the chapter, "filter $h(n)$" will be used to refer to the filter whose impulse response is $h(n)$.

Figure 7.5 shows the impulse responses of six functionally related filters. Filter $h_2(n)$ in Fig. 7.5(b) is a *sign-reversed* (i.e., reflected about the horizontal axis) version of $h_1(n)$ in Fig. 7.5(a). That is,

$$h_2(n) = -h_1(n) \tag{7.1-5}$$




a b c
d e f

**FIGURE 7.5** Six functionally related filter impulse responses: (a) reference response; (b) sign reversal; (c) and (d) order reversal (differing by the delay introduced); (e) modulation; and (f) order reversal and modulation.

Filters $h_3(n)$ and $h_4(n)$ in Figs. 7.5(c) and (d) are *order-reversed* versions of $h_1(n)$:

Order reversal is often called *time reversal* when the input sequence is a sampled analog signal.

$$h_3(n) = h_1(-n) \tag{7.1-6}$$

$$h_4(n) = h_1(K - 1 - n) \tag{7.1-7}$$

Filter $h_3(n)$ is a reflection of $h_1(n)$ about the vertical axis; filter $h_4(n)$ is a reflected and translated (i.e., shifted) version of $h_1(n)$. Neglecting translation, the responses of the two filters are identical. Filter $h_5(n)$ in Fig. 7.5(e), which is defined as

$$h_5(n) = (-1)^n h_1(n) \tag{7.1-8}$$

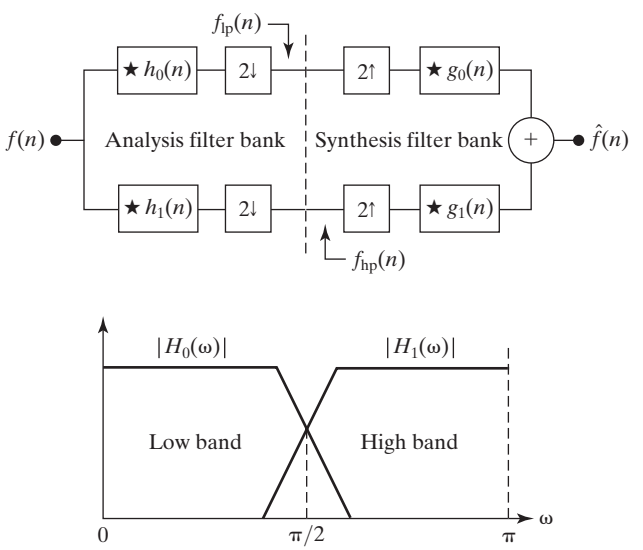
is called a *modulated* version of $h_1(n)$. Because modulation changes the signs of all odd-indexed coefficients [i.e., the coefficients for which $n$ is odd in Fig. 7.5(e)], $h_5(1) = -h_1(1)$ and $h_5(3) = -h_1(3)$, while $h_5(0) = h_1(0)$ and $h_5(2) = h_1(2)$. Finally, the sequence shown in Fig. 7.5(f) is an order-reversed version of $h_1(n)$ that is also modulated:

$$h_6(n) = (-1)^n h_1(K - 1 - n) \tag{7.1-9}$$

This sequence is included to illustrate the fact that sign reversal, order reversal, and modulation are sometimes combined in the specification of the relationship between two filters.

With this brief introduction to digital signal filtering, consider the two-band subband coding and decoding system in Fig. 7.6(a). As indicated in the figure, the system is composed of two *filter banks*, each containing two FIR filters of the type shown in Fig. 7.4(a). Note that each of the four FIR filters is depicted

A *filter bank* is a collection of two or more filters.




a
b

**FIGURE 7.6**
(a) A two-band subband coding and decoding system, and (b) its spectrum splitting properties.

as a single block in Fig. 7.6(a), with the impulse response of each filter (and the convolution symbol) written inside it. The *analysis* filter bank, which includes filters $h_0(n)$ and $h_1(n)$, is used to break input sequence $f(n)$ into two half-length sequences $f_{lp}(n)$ and $f_{hp}(n)$, the *subbands* that represent the input. Note that filters $h_0(n)$ and $h_1(n)$ are half-band filters whose idealized transfer characteristics, $H_0$ and $H_1$, are shown in Fig. 7.6(b). Filter $h_0(n)$ is a lowpass filter whose output, subband $f_{lp}(n)$, is called an *approximation* of $f(n)$; filter $h_1(n)$ is a highpass filter whose output, subband $f_{hp}(n)$, is called the high frequency or *detail* part of $f(n)$. *Synthesis* bank filters $g_0(n)$ and $g_1(n)$ combine $f_{lp}(n)$ and $f_{hp}(n)$ to produce $\hat{f}(n)$. The goal in subband coding is to select $h_0(n)$, $h_1(n)$, $g_0(n)$, and $g_1(n)$ so that $\hat{f}(n) = f(n)$. That is, so that the input and output of the subband coding and decoding system are identical. When this is accomplished, the resulting system is said to employ *perfect reconstruction filters*.

By *real-coefficient*, we mean that the filter coefficients are real (not complex) numbers.

There are many two-band, real-coefficient, FIR, perfect reconstruction filter banks described in the filter bank literature. In all of them, the synthesis filters are modulated versions of the analysis filters—with one (and only one) synthesis filter being sign reversed as well. For perfect reconstruction, the impulse responses of the synthesis and analysis filters must be related in one of the following two ways:

Equations (7.1-10) through (7.1-14) are described in detail in the filter bank literature (see, for example, Vetterli and Kovacevic [1995]).

$$\begin{aligned} g_0(n) &= (-1)^n h_1(n) \\ g_1(n) &= (-1)^{n+1} h_0(n) \end{aligned} \tag{7.1-10}$$

or

$$\begin{aligned} g_0(n) &= (-1)^{n+1} h_1(n) \\ g_1(n) &= (-1)^n h_0(n) \end{aligned} \tag{7.1-11}$$

Filters $h_0(n)$, $h_1(n)$, $g_0(n)$, and $g_1(n)$ in Eqs. (7.1-10) and (7.1-11) are said to be *cross-modulated* because diagonally opposed filters in the block diagram of Fig. 7.6(a) are related by modulation [and sign reversal when the modulation factor is $-(-1)^n$ or $(-1)^{n+1}$]. Moreover, they can be shown to satisfy the following *biorthogonality* condition:

$$\langle h_i(2n - k), g_j(k) \rangle = \delta(i - j)\delta(n), \quad i, j = \{0, 1\} \tag{7.1-12}$$

Here, $\langle h_i(2n - k), g_j(k) \rangle$ denotes the inner product of $h_i(2n - k)$ and $g_j(k)$.[†] When $i$ is not equal to $j$, the inner product is 0; when $i$ and $j$ are equal, the product is the unit discrete impulse function, $\delta(n)$. Biorthogonality will be considered again in Section 7.2.1.

Of special interest in subband coding—and in the development of the fast wavelet transform of Section 7.4—are filters that move beyond biorthogonality and require

[†]The vector inner product of sequences $f_1(n)$ and $f_2(n)$ is $\langle f_1, f_2 \rangle = \sum_n f_1^*(n) f_2(n)$, where the * denotes the complex conjugate operation. If $f_1(n)$ and $f_2(n)$ are real, $\langle f_1, f_2 \rangle = \langle f_2, f_1 \rangle$.

$$\langle g_i(n), g_j(n + 2m)\rangle = \delta(i - j)\delta(m), \quad i, j = \{0, 1\} \tag{7.1-13}$$

which defines *orthonormality* for perfect reconstruction filter banks. In addition to Eq. (7.1-13), orthonormal filters can be shown to satisfy the following two conditions:

$$\begin{aligned} g_1(n) &= (-1)^n g_0(K_{\text{even}} - 1 - n) \\ h_i(n) &= g_i(K_{\text{even}} - 1 - n), \quad i = \{0, 1\} \end{aligned} \tag{7.1-14}$$

where the subscript on $K_{\text{even}}$ is used to indicate that the number of filter coefficients must be divisible by 2 (i.e., an even number). As Eq. (7.1-14) indicates, synthesis filter $g_1$ is related to $g_0$ by order reversal and modulation. In addition, both $h_0$ and $h_1$ are order-reversed versions of synthesis filters, $g_0$ and $g_1$, respectively. Thus, an orthonormal filter bank can be developed around the impulse response of a single filter, called the *prototype*; the remaining filters can be computed from the specified prototype's impulse response. For biorthogonal filter banks, two prototypes are required; the remaining filters can be computed via Eq. (7.1-10) or (7.1-11). The generation of useful prototype filters, whether orthonormal or biorthogonal, is beyond the scope of this chapter. We simply use filters that have been presented in the literature and provide references for further study.

Before concluding the section with a 2-D subband coding example, we note that 1-D orthonormal and biorthogonal filters can be used as 2-D separable filters for the processing of images. As can be seen in Fig. 7.7, the separable filters are first applied in one dimension (e.g., vertically) and then in the other (e.g., horizontally) in the manner introduced in Section 2.6.7. Moreover, downsampling is performed in two stages—once before the second filtering operation to reduce the overall number of computations. The resulting filtered

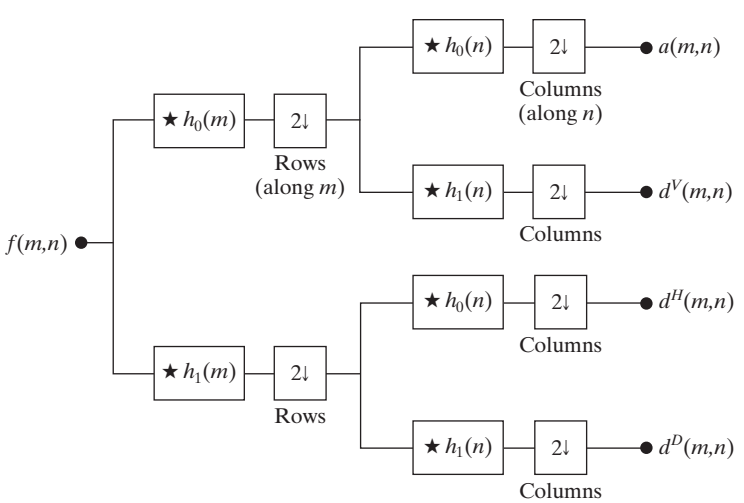



**FIGURE 7.7**
A two-dimensional, four-band filter bank for subband image coding.

outputs, denoted $a(m, n)$, $d^V(m, n)$, $d^H(m, n)$, and $d^D(m, n)$ in Fig. 7.7, are called the *approximation*, *vertical detail*, *horizontal detail*, and *diagonal detail* subbands of the input image, respectively. These subbands can be split into four smaller subbands, which can be split again, and so on—a property that will be described in greater detail in Section 7.4.

**EXAMPLE 7.2:** A four-band subband coding of the vase in Fig. 7.1.

■ Figure 7.8 shows the impulse responses of four 8-tap orthonormal filters. The coefficients of prototype synthesis filter $g_0(n)$ for $0 \leq n \leq 7$ [in Fig. 7.8(c)] are defined in Table 7.1 (Daubechies [1992]). The coefficients of the remaining orthonormal filters can be computed using Eq. (7.1-14). With the help of Fig. 7.5, note (by visual inspection) the cross modulation of the analysis and synthesis filters in Fig. 7.8. It is relatively easy to show numerically that the filters are

**TABLE 7.1** Daubechies 8-tap orthonormal filter coefficients for $g_0(n)$ (Daubechies [1992]).

| $n$ | $g_0(n)$ |
|---|---|
| 0 | 0.23037781 |
| 1 | 0.71484657 |
| 2 | 0.63088076 |
| 3 | −0.02798376 |
| 4 | −0.18703481 |
| 5 | 0.03084138 |
| 6 | 0.03288301 |
| 7 | −0.01059740 |

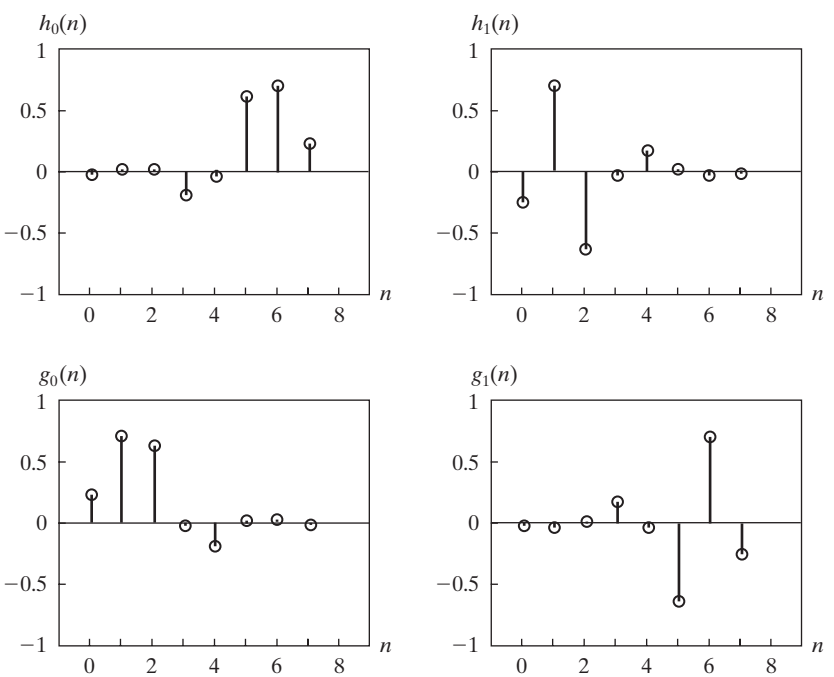
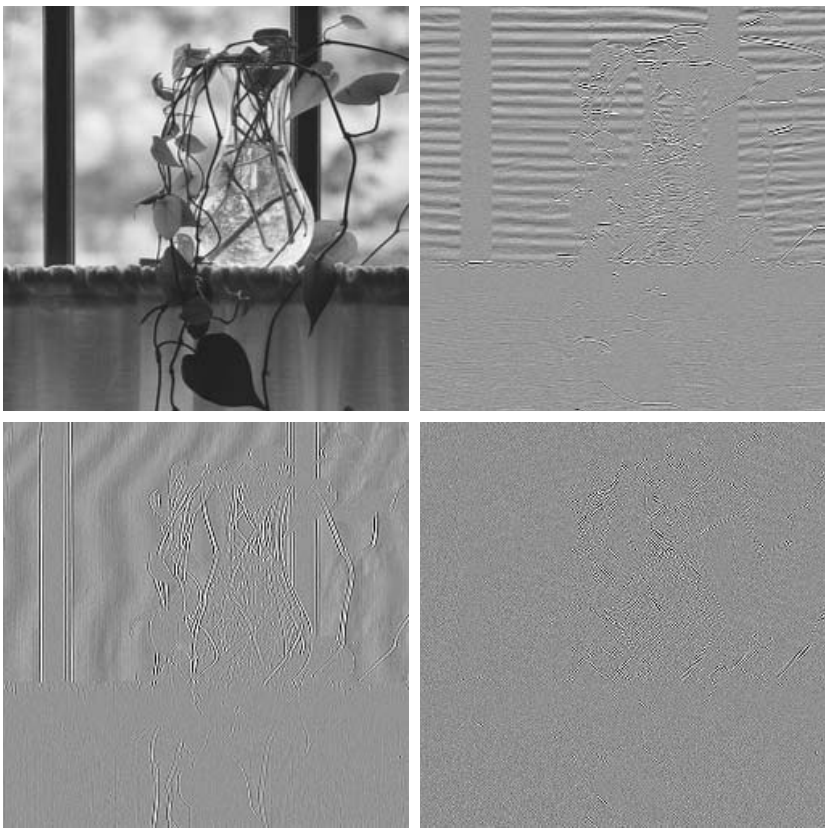



**FIGURE 7.8** The impulse responses of four 8-tap Daubechies orthonormal filters. See Table 7.1 for the values of $g_0(n)$ for $0 \leq n \leq 7$.



a b
c d

**FIGURE 7.9**
A four-band split of the vase in Fig. 7.1 using the subband coding system of Fig. 7.7. The four subbands that result are the (a) approximation, (b) horizontal detail, (c) vertical detail, and (d) diagonal detail subbands.

both biorthogonal (they satisfy Eq. 7.1-12) and orthonormal (they satisfy Eq. 7.1-13). As a result, the Daubechies 8-tap filters in Fig. 7.8 support error-free reconstruction of the decomposed input.

A four-band split of the $512 \times 512$ image of a vase in Fig. 7.1, based on the filters in Fig. 7.8, is shown in Fig. 7.9. Each quadrant of this image is a subband of size $256 \times 256$. Beginning with the upper-left corner and proceeding in a clockwise manner, the four quadrants contain approximation subband $a$, horizontal detail subband $d^H$, diagonal detail subband $d^D$, and vertical detail subband $d^V$, respectively. All subbands, except the approximation subband in Fig. 7.9(a), have been scaled to make their underlying structure more visible. Note the visual effects of aliasing that are present in Figs. 7.9(b) and (c)—the $d^H$ and $d^V$ subbands. The wavy lines in the window area are due to the downsampling of a barely discernable window screen in Fig. 7.1. Despite the aliasing, the original image can be reconstructed from the subbands in Fig. 7.9 without error. The required synthesis filters, $g_0(n)$ and $g_1(n)$, are determined from Table 7.1 and Eq. (7.1-14), and incorporated into a filter bank that roughly mirrors the system in Fig. 7.7. In the new filter bank, filters $h_i(n)$ for $i = \{0, 1\}$ are replaced by their $g_i(n)$ counterparts, and upsamplers and summers are added. ■

See Section 4.5.4 for more on aliasing.

### 7.1.3 The Haar Transform

The third and final imaging-related operation with ties to multiresolution analysis that we will look at is the Haar transform (Haar [1910]). Within the context of this chapter, its importance stems from the fact that its basis functions (defined below) are the oldest and simplest known orthonormal wavelets. They will be used in a number of examples in the sections that follow.

With reference to the discussion in Section 2.6.7, the Haar transform can be expressed in the following matrix form

$$\mathbf{T} = \mathbf{HFH}^T \tag{7.1-15}$$

where $\mathbf{F}$ is an $N \times N$ image matrix, $\mathbf{H}$ is an $N \times N$ Haar transformation matrix, and $\mathbf{T}$ is the resulting $N \times N$ transform. The transpose is required because $\mathbf{H}$ is not symmetric; in Eq. (2.6-38) of Section 2.6.7, the transformation matrix is assumed to be symmetric. For the Haar transform, $\mathbf{H}$ contains the Haar basis functions, $h_k(z)$. They are defined over the continuous, closed interval $z \in [0, 1]$ for $k = 0, 1, 2, \ldots, N - 1$, where $N = 2^n$. To generate $\mathbf{H}$, we define the integer $k$ such that $k = 2^p + q - 1$, where $0 \le p \le n - 1$, $q = 0$ or 1 for $p = 0$, and $1 \le q \le 2^p$ for $p \ne 0$. Then the *Haar basis functions* are

$$h_0(z) = h_{00}(z) = \frac{1}{\sqrt{N}}, \quad z \in [0, 1] \tag{7.1-16}$$

and

$$h_k(z) = h_{pq}(z) = \frac{1}{\sqrt{N}} \begin{cases} 2^{p/2} & (q - 1)/2^p \le z < (q - 0.5)/2^p \\ -2^{p/2} & (q - 0.5)/2^p \le z < q/2^p \\ 0 & \text{otherwise, } z \in [0, 1] \end{cases} \tag{7.1-17}$$

The $i$th row of an $N \times N$ Haar transformation matrix contains the elements of $h_i(z)$ for $z = 0/N, 1/N, 2/N, \ldots, (N - 1)/N$. For instance, if $N = 2$, the first row of the $2 \times 2$ Haar matrix is computed using $h_0(z)$ with $z = 0/2, 1/2$. From Eq. (7.1-16), $h_0(z)$ is equal to $1/\sqrt{2}$, independent of $z$, so the first row of $\mathbf{H}_2$ has two identical $1/\sqrt{2}$ elements. The second row is obtained by computing $h_1(z)$ for $z = 0/2, 1/2$. Because $k = 2^p + q - 1$, when $k = 1$, $p = 0$ and $q = 1$. Thus, from Eq. (7.1-17), $h_1(0) = 2^0/\sqrt{2} = 1/\sqrt{2}$, $h_1(1/2) = -2^0/\sqrt{2} = -1/\sqrt{2}$, and the $2 \times 2$ Haar matrix is

$$\mathbf{H}_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{7.1-18}$$

If $N = 4$, $k$, $q$, and $p$ assume the values

| $k$ | $p$ | $q$ |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 2 | 1 | 1 |
| 3 | 1 | 2 |

and the $4 \times 4$ transformation matrix, $\mathbf{H}_4$, is

$$\mathbf{H}_4 = \frac{1}{\sqrt{4}} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ \sqrt{2} & -\sqrt{2} & 0 & 0 \\ 0 & 0 & \sqrt{2} & -\sqrt{2} \end{bmatrix} \tag{7.1-19}$$

Our principal interest in the Haar transform is that the rows of $\mathbf{H}_2$ can be used to define the analysis filters, $h_0(n)$ and $h_1(n)$, of a 2-tap perfect reconstruction filter bank (see the previous section), as well as the scaling and wavelet vectors (defined in Sections 7.2.2 and 7.2.3, respectively) of the simplest and oldest wavelet transform (see Example 7.10 in Section 7.4). Rather than concluding the section with the computation of a Haar transform, we close with an example that illustrates the influence of the decomposition methods that have been considered to this point on the methods that will be developed in the remainder of the chapter.

**EXAMPLE 7.3:** Haar functions in a discrete wavelet transform.

■ Figure 7.10(a) shows a decomposition of the $512 \times 512$ image in Fig. 7.1 that combines the key features of pyramid coding, subband coding, and the Haar transform (the three techniques we have discussed so far). Called the discrete wavelet transform (and developed later in the chapter), the representation is characterized by the following important features:

1. With the exception of the subimage in the upper-left corner of Fig. 7.10(a), the local histograms are very similar. Many of the pixels are close to zero. Because the subimages (except for the subimage in the upper-left corner) have been scaled to make their underlying structure more visible, the displayed histograms are peaked at intensity 128 (the zeroes have been scaled to mid-gray). The large number of zeroes in the decomposition makes the image an excellent candidate for compression (see Chapter 8).
2. In a manner that is similar to the way in which the levels of the prediction residual pyramid of Fig. 7.3(b) were used to create approximation images of differing resolutions, the subimages in Fig. 7.10(a) can be used to construct both coarse and fine resolution approximations of the original vase image in Fig. 7.1. Figures 7.10(b) through (d), which are of size

a
b c d

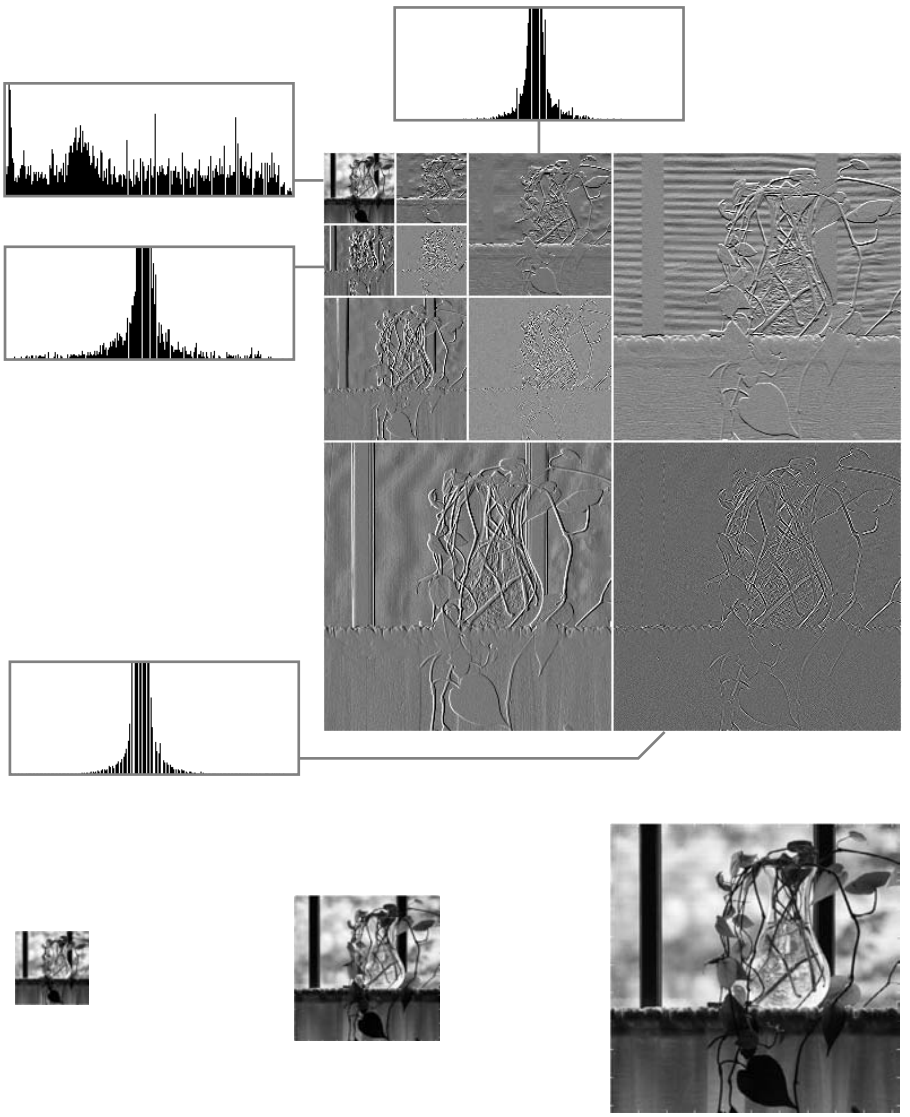


**FIGURE 7.10**
(a) A discrete wavelet transform using Haar $\mathbf{H}_2$ basis functions. Its local histogram variations are also shown. (b)–(d) Several different approximations ($64 \times 64$, $128 \times 128$, and $256 \times 256$) that can be obtained from (a).

$64 \times 64$, $128 \times 128$, and $256 \times 256$, respectively, were generated from the subimages in Fig. 7.10(a). A perfect $512 \times 512$ reconstruction of the original image is also possible.

3. Like the subband coding decomposition in Fig. 7.9, a simple real-coefficient, FIR filter bank of the form given in Fig. 7.7 was used to produce Fig. 7.10(a). After the generation of a four subband image like that of Fig. 7.9, the $256 \times 256$ approximation subband was decomposed and replaced by four $128 \times 128$ subbands (using the same filter bank), and the resulting approximation subband was again decomposed and replaced by four $64 \times 64$ subbands. This process produced the unique arrangement of subimages that

characterizes discrete wavelet transforms. The subimages in Fig. 7.10(a) become smaller in size as you move from the lower-right-hand to upper-left-hand corner of the image.

4. Figure 7.10(a) is not the Haar transform of the image in Fig. 7.1. Although the filter bank coefficients that were used to produce this decomposition were taken from Haar transformation matrix $\mathbf{H}_2$, a variety of othronormal and biorthogonal filter bank coefficients can be used in discrete wavelet transforms.
5. As will be shown in Section 7.4, each subimage in Fig. 7.10(a) represents a specific band of spatial frequencies in the original image. In addition, many of the subimages demonstrate directional sensitivity [e.g., the subimage in the upper-right corner of Fig. 7.10(a) captures horizontal edge information in the original image].

Considering this impressive list of features, it is remarkable that the discrete wavelet transform of Fig. 7.10(a) was generated using two 2-tap digital filters with a total of four filter coefficients. ■

## 7.2 Multiresolution Expansions

The previous section introduced three well-known imaging techniques that play an important role in a mathematical framework called *multiresolution analysis* (MRA). In MRA, a *scaling function* is used to create a series of approximations of a function or image, each differing by a factor of 2 in resolution from its nearest neighboring approximations. Additional functions, called *wavelets*, are then used to encode the difference in information between adjacent approximations.

### 7.2.1 Series Expansions

A signal or function $f(x)$ can often be better analyzed as a linear combination of expansion functions

$$f(x) = \sum_k \alpha_k \varphi_k(x) \tag{7.2-1}$$

where $k$ is an integer index of a finite or infinite sum, the $\alpha_k$ are real-valued *expansion coefficients*, and the $\varphi_k(x)$ are real-valued *expansion functions*. If the expansion is unique—that is, there is only one set of $\alpha_k$ for any given $f(x)$—the $\varphi_k(x)$ are called *basis functions*, and the *expansion set*, $\{\varphi_k(x)\}$, is called a *basis* for the class of functions that can be so expressed. The expressible functions form a *function space* that is referred to as the *closed span* of the expansion set, denoted

$$V = \overline{\underset{k}{\text{Span}}\{\varphi_k(x)\}} \tag{7.2-2}$$

To say that $f(x) \in V$ means that $f(x)$ is in the closed span of $\{\varphi_k(x)\}$ and can be written in the form of Eq. (7.2-1).

For any function space $V$ and corresponding expansion set $\{\varphi_k(x)\}$, there is a set of *dual* functions denoted $\{\tilde{\varphi}_k(x)\}$ that can be used to compute the $\alpha_k$ coefficients of Eq. (7.2-1) for any $f(x) \in V$. These coefficients are computed by taking the *integral inner products*[†] of the dual $\tilde{\varphi}_k(x)$ and function $f(x)$. That is,

$$\alpha_k = \langle \tilde{\varphi}_k(x), f(x) \rangle = \int \tilde{\varphi}_k^*(x) f(x)\, dx \tag{7.2-3}$$

where the * denotes the complex conjugate operation. Depending on the orthogonality of the expansion set, this computation assumes one of three possible forms. Problem 7.10 at the end of the chapter illustrates the three cases using vectors in two-dimensional Euclidean space.

*Case 1:* If the expansion functions form an orthonormal basis for $V$, meaning that

$$\langle \varphi_j(x), \varphi_k(x) \rangle = \delta_{jk} = \begin{cases} 0 & j \neq k \\ 1 & j = k \end{cases} \tag{7.2-4}$$

the basis and its dual are equivalent. That is, $\varphi_k(x) = \tilde{\varphi}_k(x)$ and Eq. (7.2-3) becomes

$$\alpha_k = \langle \varphi_k(x), f(x) \rangle \tag{7.2-5}$$

The $\alpha_k$ are computed as the inner products of the basis functions and $f(x)$.

*Case 2:* If the expansion functions are not orthonormal, but are an orthogonal basis for $V$, then

$$\langle \varphi_j(x), \varphi_k(x) \rangle = 0 \quad j \neq k \tag{7.2-6}$$

and the basis functions and their duals are called *biorthogonal*. The $\alpha_k$ are computed using Eq. (7.2-3), and the biorthogonal basis and its dual are such that

$$\langle \varphi_j(x), \tilde{\varphi}_k(x) \rangle = \delta_{jk} = \begin{cases} 0 & j \neq k \\ 1 & j = k \end{cases} \tag{7.2-7}$$

*Case 3:* If the expansion set is not a basis for $V$, but supports the expansion defined in Eq. (7.2-1), it is a spanning set in which there is more than one set of $\alpha_k$ for any $f(x) \in V$. The expansion functions and their duals are said to be *overcomplete* or redundant. They form a *frame* in which[‡]

$$A\|f(x)\|^2 \leq \sum_k |\langle \varphi_k(x), f(x) \rangle|^2 \leq B\|f(x)\|^2 \tag{7.2-8}$$

[†]The integral inner product of two real or complex-valued functions $f(x)$ and $g(x)$ is $\langle f(x), g(x) \rangle = \int f^*(x) g(x)\, dx$. If $f(x)$ is real, $f^*(x) = f(x)$ and $\langle f(x), g(x) \rangle = \int f(x) g(x)\, dx$.

[‡]The norm of $f(x)$, denoted $\|f(x)\|$, is defined as the square root of the absolute value of the inner product of $f(x)$ with itself.

for some $A > 0$, $B < \infty$, and all $f(x) \in V$. Dividing this equation by the norm squared of $f(x)$, we see that $A$ and $B$ "frame" the normalized inner products of the expansion coefficients and the function. Equations similar to (7.2-3) and (7.2-5) can be used to find the expansion coefficients for frames. If $A = B$, the expansion set is called a *tight frame* and it can be shown that (Daubechies [1992])

$$f(x) = \frac{1}{A} \sum_k \langle \varphi_k(x), f(x) \rangle \varphi_k(x) \tag{7.2-9}$$

Except for the $A^{-1}$ term, which is a measure of the frame's redundancy, this is identical to the expression obtained by substituting Eq. (7.2-5) (for orthonormal bases) into Eqs. (7.2-1).

### 7.2.2 Scaling Functions

Consider the set of expansion functions composed of integer translations and binary scalings of the real, square-integrable function $\varphi(x)$; this is the set $\{\varphi_{j,k}(x)\}$, where

$$\varphi_{j,k}(x) = 2^{j/2} \varphi(2^j x - k) \tag{7.2-10}$$

for all $j, k \in \mathbf{Z}$ and $\varphi(x) \in L^2(\mathbf{R})$.[†] Here, $k$ determines the position of $\varphi_{j,k}(x)$ along the $x$-axis, and $j$ determines the width of $\varphi_{j,k}(x)$—that is, how broad or narrow it is along the $x$-axis. The term $2^{j/2}$ controls the amplitude of the function. Because the shape of $\varphi_{j,k}(x)$ changes with $j$, $\varphi(x)$ is called a *scaling function*. By choosing $\varphi(x)$ properly, $\{\varphi_{j,k}(x)\}$ can be made to span $L^2(\mathbf{R})$, which is the set of all measurable, square-integrable functions.

If we restrict $j$ in Eq. (7.2-10) to a specific value, say $j = j_0$, the resulting expansion set, $\{\varphi_{j_0,k}(x)\}$, is a subset of $\{\varphi_{j,k}(x)\}$ that spans a subspace of $L^2(\mathbf{R})$. Using the notation of the previous section, we can define that subspace as

$$V_{j_0} = \overline{\underset{k}{\text{Span}}\{\varphi_{j_0,k}(x)\}} \tag{7.2-11}$$

That is, $V_{j_0}$ is the span of $\varphi_{j_0,k}(x)$ over $k$. If $f(x) \in V_{j_0}$, we can write

$$f(x) = \sum_k \alpha_k \varphi_{j_0,k}(x) \tag{7.2-12}$$

More generally, we will denote the subspace spanned over $k$ for any $j$ as

$$V_j = \overline{\underset{k}{\text{Span}}\{\varphi_{j,k}(x)\}} \tag{7.2-13}$$

As will be seen in the following example, increasing $j$ increases the size of $V_j$, allowing functions with smaller variations or finer detail to be included in the subspace. This is a consequence of the fact that, as $j$ increases, the $\varphi_{j,k}(x)$ that are used to represent the subspace functions become narrower and separated by smaller changes in $x$.

[†]The notation $L^2(\mathbf{R})$, where $\mathbf{R}$ is the set of real numbers, denotes the set of measurable, square-integrable, one-dimensional functions; $\mathbf{Z}$ is the set of integers.

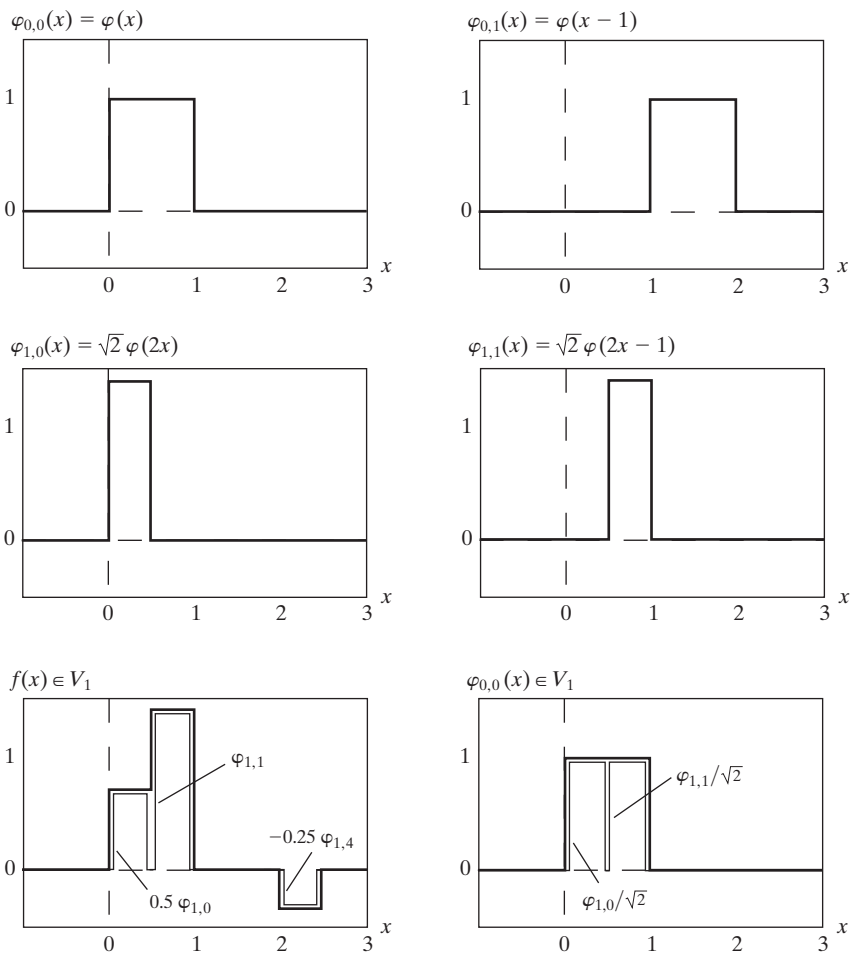
**EXAMPLE 7.4:** The Haar scaling function.

■ Consider the unit-height, unit-width scaling function (Haar [1910])

$$\varphi(x) = \begin{cases} 1 & 0 \le x < 1 \\ 0 & \text{otherwise} \end{cases} \tag{7.2-14}$$

Figures 7.11(a) through (d) show four of the many expansion functions that can be generated by substituting this pulse-shaped scaling function into Eq. (7.2-10). Note that the expansion functions for $j = 1$ in Figs. 7.11(c) and (d) are half as wide as those for $j = 0$ in Figs. 7.11(a) and (b). For a given interval on $x$, we can define twice as many $V_1$ scaling functions as $V_0$ scaling functions (e.g., $\varphi_{1,0}$ and $\varphi_{1,1}$ of $V_1$ versus $\varphi_{0,0}$ of $V_0$ for the interval $0 \le x < 1$).

Figure 7.11(e) shows a member of subspace $V_1$. This function does not belong to $V_0$, because the $V_0$ expansion functions in 7.11(a) and (b) are too coarse to represent it. Higher-resolution functions like those in 7.11(c) and (d)




a b
c d
e f

**FIGURE 7.11** Some Haar scaling functions.

are required. They can be used, as shown in (e), to represent the function by the three-term expansion

$$f(x) = 0.5\varphi_{1,0}(x) + \varphi_{1,1}(x) - 0.25\varphi_{1,4}(x)$$

To conclude the example, Fig. 7.11(f) illustrates the decomposition of $\varphi_{0,0}(x)$ as a sum of $V_1$ expansion functions. In a similar manner, any $V_0$ expansion function can be decomposed using

$$\varphi_{0,k}(x) = \frac{1}{\sqrt{2}}\varphi_{1,2k}(x) + \frac{1}{\sqrt{2}}\varphi_{1,2k+1}(x)$$

Thus, if $f(x)$ is an element of $V_0$, it is also an element of $V_1$. This is because all $V_0$ expansion functions are contained in $V_1$. Mathematically, we write that $V_0$ is a subspace of $V_1$, denoted $V_0 \subset V_1$. ■

The simple scaling function in the preceding example obeys the four fundamental requirements of multiresolution analysis (Mallat [1989a]):

*MRA Requirement 1: The scaling function is orthogonal to its integer translates.*

This is easy to see in the case of the Haar function, because whenever it has a value of 1, its integer translates are 0, so that the product of the two is 0. The Haar scaling function is said to have *compact support*, which means that it is 0 everywhere outside a finite interval called the *support*. In fact, the width of the support is 1; it is 0 outside the half open interval [0, 1). It should be noted that the requirement for orthogonal integer translates becomes harder to satisfy as the width of support of the scaling function becomes larger than 1.

*MRA Requirement 2: The subspaces spanned by the scaling function at low scales are nested within those spanned at higher scales.*

As can be seen in Fig. 7.12, subspaces containing high-resolution functions must also contain all lower resolution functions. That is,

$$V_{-\infty} \subset \cdots \subset V_{-1} \subset V_0 \subset V_1 \subset V_2 \subset \cdots \subset V_{\infty} \qquad (7.2\text{-}15)$$

Moreover, the subspaces satisfy the intuitive condition that if $f(x) \in V_j$, then $f(2x) \in V_{j+1}$. The fact that the Haar scaling function meets this requirement

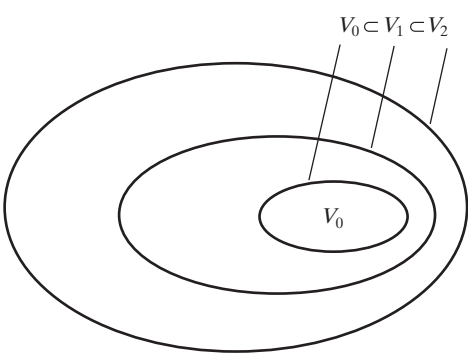



**FIGURE 7.12** The nested function spaces spanned by a scaling function.

should not be taken to indicate that any function with a support width of 1 automatically satisfies the condition. It is left as an exercise for the reader to show that the equally simple function

$$\varphi(x) = \begin{cases} 1 & 0.25 \le x < 0.75 \\ 0 & \text{elsewhere} \end{cases}$$

is not a valid scaling function for a multiresolution analysis (see Problem 7.11).

*MRA Requirement 3: The only function that is common to all $V_j$ is $f(x) = 0$.* If we consider the coarsest possible expansion functions (i.e., $j = -\infty$), the only representable function is the function of no information. That is,

$$V_{-\infty} = \{0\} \tag{7.2-16}$$

*MRA Requirement 4: Any function can be represented with arbitrary precision.* Though it may not be possible to expand a particular $f(x)$ at an arbitrarily coarse resolution, as was the case for the function in Fig. 7.11(e), all measurable, square-integrable functions can be represented by the scaling functions in the limit as $j \to \infty$. That is,

$$V_{\infty} = \left\{L^2(\mathbf{R})\right\} \tag{7.2-17}$$

Under these conditions, the expansion functions of subspace $V_j$ can be expressed as a weighted sum of the expansion functions of subspace $V_{j+1}$. Using Eq. (7.2-12), we let

$$\varphi_{j,k}(x) = \sum_n \alpha_n \varphi_{j+1,n}(x)$$

where the index of summation has been changed to $n$ for clarity. Substituting for $\varphi_{j+1,n}(x)$ from Eq. (7.2-10) and changing variable $\alpha_n$ to $h_\varphi(n)$, this becomes

The $\alpha_n$ are changed to $h_\varphi(n)$ because they are used later (see Section 7.4) as filter bank coefficients.

$$\varphi_{j,k}(x) = \sum_n h_\varphi(n) 2^{(j+1)/2} \varphi(2^{j+1}x - n)$$

Because $\varphi(x) = \varphi_{0,0}(x)$, both $j$ and $k$ can be set to 0 to obtain the simpler nonsubscripted expression

$$\varphi(x) = \sum_n h_\varphi(n) \sqrt{2} \varphi(2x - n) \tag{7.2-18}$$

The $h_\varphi(n)$ coefficients in this recursive equation are called *scaling function coefficients*; $h_\varphi$ is referred to as a *scaling vector*. Equation (7.2-18) is fundamental to multiresolution analysis and is called the *refinement equation*, the *MRA equation*, or the *dilation equation*. It states that the expansion functions of any subspace can be built from double-resolution copies of themselves—that is, from expansion functions of the next higher resolution space. The choice of a reference subspace, $V_0$, is arbitrary.

**EXAMPLE 7.5:** Haar scaling function coefficients.

■ The scaling function coefficients for the Haar function of Eq. (7.2-14) are $h_\varphi(0) = h_\varphi(1) = 1/\sqrt{2}$, the first row of matrix $\mathbf{H}_2$ in Eq. (7.1-18). Thus, Eq. (7.2-18) yields

$$\varphi(x) = \frac{1}{\sqrt{2}}\left[\sqrt{2}\varphi(2x)\right] + \frac{1}{\sqrt{2}}\left[\sqrt{2}\varphi(2x-1)\right]$$

This decomposition was illustrated graphically for $\varphi_{0,0}(x)$ in Fig. 7.11(f), where the bracketed terms of the preceding expression are seen to be $\varphi_{1,0}(x)$ and $\varphi_{1,1}(x)$. Additional simplification yields $\varphi(x) = \varphi(2x) + \varphi(2x-1)$. ■

### 7.2.3 Wavelet Functions

Given a scaling function that meets the MRA requirements of the previous section, we can define a *wavelet function* $\psi(x)$ that, together with its integer translates and binary scalings, spans the difference between any two adjacent scaling subspaces, $V_j$ and $V_{j+1}$. The situation is illustrated graphically in Fig. 7.13. We define the set $\{\psi_{j,k}(x)\}$ of wavelets

$$\psi_{j,k}(x) = 2^{j/2}\psi(2^j x - k) \tag{7.2-19}$$

for all $k \in \mathbf{Z}$ that span the $W_j$ spaces in the figure. As with scaling functions, we write

$$W_j = \overline{\operatorname{Span}_k\{\psi_{j,k}(x)\}} \tag{7.2-20}$$

and note that if $f(x) \in W_j$,

$$f(x) = \sum_k \alpha_k \psi_{j,k}(x) \tag{7.2-21}$$

The scaling and wavelet function subspaces in Fig. 7.13 are related by

$$V_{j+1} = V_j \oplus W_j \tag{7.2-22}$$

where $\oplus$ denotes the union of spaces (like the union of sets). The orthogonal complement of $V_j$ in $V_{j+1}$ is $W_j$, and all members of $V_j$ are orthogonal to the members of $W_j$. Thus,

$$\langle \varphi_{j,k}(x), \psi_{j,l}(x) \rangle = 0 \tag{7.2-23}$$

for all appropriate $j, k, l \in \mathbf{Z}$.

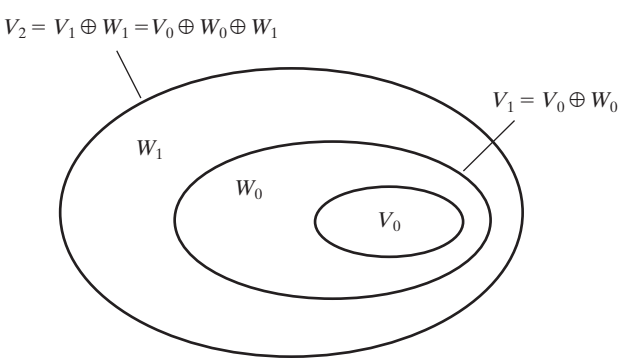



**FIGURE 7.13** The relationship between scaling and wavelet function spaces.

We can now express the space of all measurable, square-integrable functions as

$$L^2(\mathbf{R}) = V_0 \oplus W_0 \oplus W_1 \oplus \ldots \tag{7.2-24}$$

or

$$L^2(\mathbf{R}) = V_1 \oplus W_1 \oplus W_2 \oplus \ldots \tag{7.2-25}$$

or even

$$L^2(\mathbf{R}) = \cdots \oplus W_{-2} \oplus W_{-1} \oplus W_0 \oplus W_1 \oplus W_2 \oplus \ldots \tag{7.2-26}$$

which eliminates the scaling function, and represents a function in terms of wavelets alone [i.e., there are only wavelet function spaces in Eq. (7.2-26)]. Note that if $f(x)$ is an element of $V_1$, but not $V_0$, an expansion using Eq. (7.2-24) contains an *approximation* of $f(x)$ using $V_0$ scaling functions. Wavelets from $W_0$ would encode the *difference* between this approximation and the actual function. Equations (7.2-24) through (7.2-26) can be generalized to yield

$$L^2(\mathbf{R}) = V_{j_0} \oplus W_{j_0} \oplus W_{j_0+1} \oplus \ldots \tag{7.2-27}$$

where $j_0$ is an arbitrary starting scale.

Since wavelet spaces reside within the spaces spanned by the next higher resolution scaling functions (see Fig. 7.13), any wavelet function—like its scaling function counterpart of Eq. (7.2-18)—can be expressed as a weighted sum of shifted, double-resolution scaling functions. That is, we can write

$$\psi(x) = \sum_n h_\psi(n)\sqrt{2}\varphi(2x - n) \tag{7.2-28}$$

where the $h_\psi(n)$ are called the *wavelet function coefficients* and $h_\psi$ is the *wavelet vector*. Using the condition that wavelets span the orthogonal complement spaces in Fig. 7.13 and that integer wavelet translates are orthogonal, it can be shown that $h_\psi(n)$ is related to $h_\varphi(n)$ by (see, for example, Burrus, Gopinath, and Guo [1998])

$$h_\psi(n) = (-1)^n h_\varphi(1 - n) \tag{7.2-29}$$

Note the similarity of this result and Eq. (7.1-14), the relationship governing the impulse responses of orthonormal subband coding and decoding filters.

**EXAMPLE 7.6:** The Haar wavelet function coefficients.

■ In the previous example, the Haar scaling vector was defined as $h_\varphi(0) = h_\varphi(1) = 1/\sqrt{2}$. Using Eq. (7.2-29), the corresponding wavelet vector is $h_\psi(0) = (-1)^0 h_\varphi(1 - 0) = 1/\sqrt{2}$ and $h_\psi(1) = (-1)^1 h_\varphi(1 - 1) = -1/\sqrt{2}$. Note that these coefficients correspond to the second row of matrix $\mathbf{H}_2$ in Eq. (7.1-18). Substituting these values into Eq. (7.2-28), we get

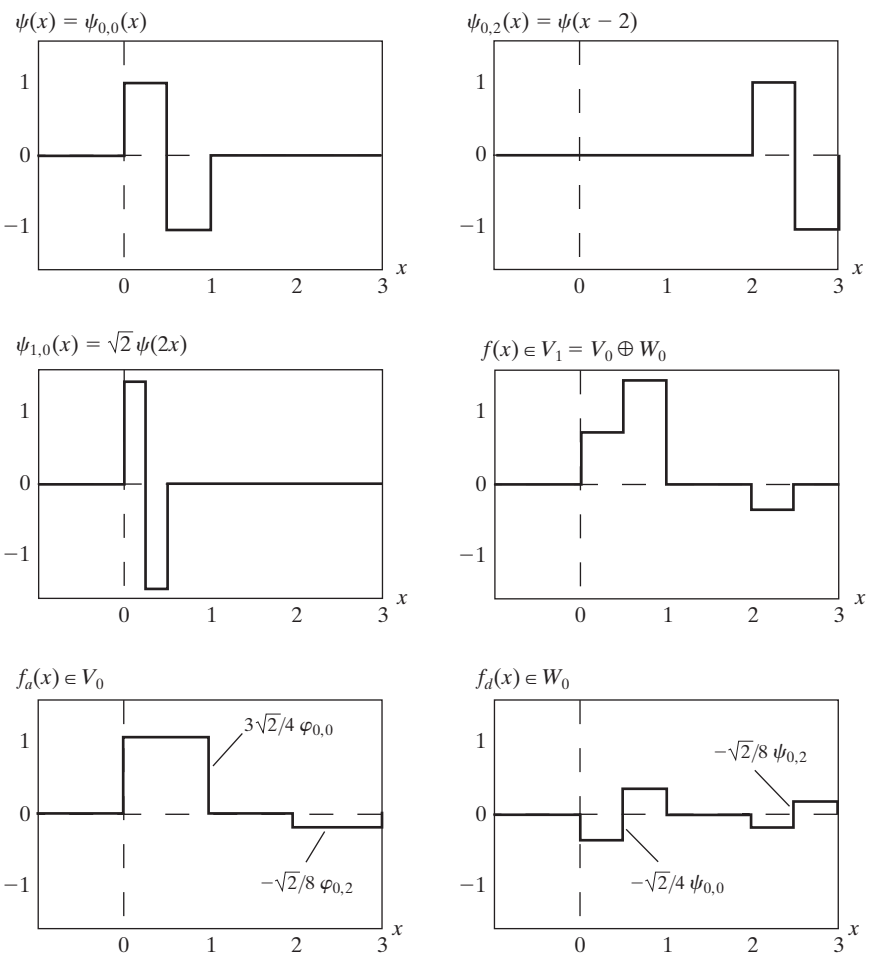
$\psi(x) = \varphi(2x) - \varphi(2x - 1)$, which is plotted in Fig. 7.14(a). Thus, the Haar wavelet function is

$$\psi(x) = \begin{cases} 1 & 0 \le x < 0.5 \\ -1 & 0.5 \le x < 1 \\ 0 & \text{elsewhere} \end{cases} \tag{7.2-30}$$

Using Eq. (7.2-19), we can now generate the universe of scaled and translated Haar wavelets. Two such wavelets, $\psi_{0,2}(x)$ and $\psi_{1,0}(x)$, are plotted in Figs. 7.14(b) and (c), respectively. Note that wavelet $\psi_{1,0}(x)$ for space $W_1$ is narrower than $\psi_{0,2}(x)$ for $W_0$; it can be used to represent finer detail.

Figure 7.14(d) shows a function of subspace $V_1$ that is not in subspace $V_0$. This function was considered in an earlier example [see Fig. 7.11(e)]. Although the function cannot be represented accurately in $V_0$, Eq. (7.2-22) indicates that it can be expanded using $V_0$ and $W_0$ expansion functions. The resulting expansion is

$$f(x) = f_a(x) + f_d(x)$$




a b
c d
e f

**FIGURE 7.14** Haar wavelet functions in $W_0$ and $W_1$.

where

$$f_a(x) = \frac{3\sqrt{2}}{4}\varphi_{0,0}(x) - \frac{\sqrt{2}}{8}\varphi_{0,2}(x)$$

and

$$f_d(x) = \frac{-\sqrt{2}}{4}\psi_{0,0}(x) - \frac{\sqrt{2}}{8}\psi_{0,2}(x)$$

Here, $f_a(x)$ is an approximation of $f(x)$ using $V_0$ scaling functions, while $f_d(x)$ is the difference $f(x) - f_a(x)$ as a sum of $W_0$ wavelets. The two expansions, which are shown in Figs. 7.14(e) and (f), divide $f(x)$ in a manner similar to a lowpass and highpass filter as discussed in connection with Fig. 7.6. The low frequencies of $f(x)$ are captured in $f_a(x)$—it assumes the average value of $f(x)$ in each integer interval—while the high-frequency details are encoded in $f_d(x)$. ■

## 7.3 Wavelet Transforms in One Dimension

We can now formally define several closely related wavelet transformations: the generalized *wavelet series expansion*, the *discrete wavelet transform*, and the *continuous wavelet transform*. Their counterparts in the Fourier domain are the Fourier series expansion, the discrete Fourier transform, and the integral Fourier transform, respectively. In Section 7.4, we develop a computationally efficient implementation of the discrete wavelet transform called the *fast wavelet transform*.

### 7.3.1 The Wavelet Series Expansions

We begin by defining the *wavelet series expansion* of function $f(x) \in L^2(\mathbf{R})$ relative to wavelet $\psi(x)$ and scaling function $\varphi(x)$. In accordance with Eq. (7.2-27), $f(x)$ can be represented by a scaling function expansion in subspace $V_{j_0}$ [Eq. (7.2-12) defines such an expansion] and some number of wavelet function expansions in subspaces $W_{j_0}, W_{j_0+1}, \ldots$ [as defined in Eq. (7.2-21)]. Thus,

$$f(x) = \sum_k c_{j_0}(k)\varphi_{j_0,k}(x) + \sum_{j=j_0}^{\infty}\sum_k d_j(k)\psi_{j,k}(x) \tag{7.3-1}$$

where $j_0$ is an arbitrary starting scale and the $c_{j_0}(k)$ and $d_j(k)$ are relabeled $\alpha_k$ from Eqs. (7.2-12) and (7.2-21), respectively. The $c_{j_0}(k)$ are normally called *approximation* and/or *scaling coefficients*; the $d_j(k)$ are referred to as *detail* and/or *wavelet coefficients*. This is because the first sum in Eq. (7.3-1) uses scaling functions to provide an approximation of $f(x)$ at scale $j_0$ [unless $f(x) \in V_{j_0}$ so that the sum of the scaling functions is equal to $f(x)$]. For each higher scale $j \geq j_0$ in the second sum, a finer resolution function—a sum of wavelets—is added to the approximation to provide increasing detail. If the expansion

functions form an orthonormal basis or tight frame, which is often the case, the expansion coefficients are calculated—based on Eqs. (7.2-5) and (7.2-9)—as

$$c_{j_0}(k) = \langle f(x), \varphi_{j_0,k}(x) \rangle = \int f(x)\varphi_{j_0,k}(x)\,dx \tag{7.3-2}$$

Because $f$ is real, no conjugates are needed in the inner products of Eqs. (7.3-2) and (7.3-3).

and

$$d_j(k) = \langle f(x), \psi_{j,k}(x) \rangle = \int f(x)\psi_{j,k}(x)\,dx \tag{7.3-3}$$

In Eqs. (7.2-5) and (7.2-9), the expansion coefficients (i.e., the $\alpha_k$) are defined as inner products of the function being expanded and the expansion functions being used. In Eqs. (7.3-2) and (7.3-3), the expansion functions are the $\varphi_{j_0,k}$ and $\psi_{j,k}$; the expansion coefficients are the $c_{j_0}$ and $d_j$. If the expansion functions are part of a biorthogonal basis, the $\varphi$ and $\psi$ terms in these equations must be replaced by their dual functions, $\widetilde{\varphi}$ and $\widetilde{\psi}$, respectively.

**EXAMPLE 7.7:** The Haar wavelet series expansion of $y = x^2$.

■ Consider the simple function

$$y = \begin{cases} x^2 & 0 \le x \le 1 \\ 0 & \text{otherwise} \end{cases}$$

shown in Fig. 7.15(a). Using Haar wavelets—see Eqs. (7.2-14) and (7.2-30)—and a starting scale $j_0 = 0$, Eqs. (7.3-2) and (7.3-3) can be used to compute the following expansion coefficients:

$$c_0(0) = \int_0^1 x^2\varphi_{0,0}(x)\,dx = \int_0^1 x^2\,dx = \frac{x^3}{3}\bigg|_0^1 = \frac{1}{3}$$

$$d_0(0) = \int_0^1 x^2\psi_{0,0}(x)\,dx = \int_0^{0.5} x^2\,dx - \int_{0.5}^1 x^2\,dx = -\frac{1}{4}$$

$$d_1(0) = \int_0^1 x^2\psi_{1,0}(x)\,dx = \int_0^{0.25} x^2\sqrt{2}\,dx - \int_{0.25}^{0.5} x^2\sqrt{2}\,dx = -\frac{\sqrt{2}}{32}$$

$$d_1(1) = \int_0^1 x^2\psi_{1,1}(x)\,dx = \int_{0.5}^{0.75} x^2\sqrt{2}\,dx - \int_{0.75}^1 x^2\sqrt{2}\,dx = -\frac{3\sqrt{2}}{32}$$

Substituting these values into Eq. (7.3-1), we get the wavelet series expansion

$$y = \underbrace{\underbrace{\frac{1}{3}\varphi_{0,0}(x)}_{V_0} + \underbrace{\left[-\frac{1}{4}\psi_{0,0}(x)\right]}_{W_0}}_{V_1 = V_0 \oplus W_0} + \underbrace{\left[-\frac{\sqrt{2}}{32}\psi_{1,0}(x) - \frac{3\sqrt{2}}{32}\psi_{1,1}(x)\right]}_{W_1} + \cdots$$

$$V_2 = V_1 \oplus W_1 = V_0 \oplus W_0 \oplus W_1$$

a b
c d
e f

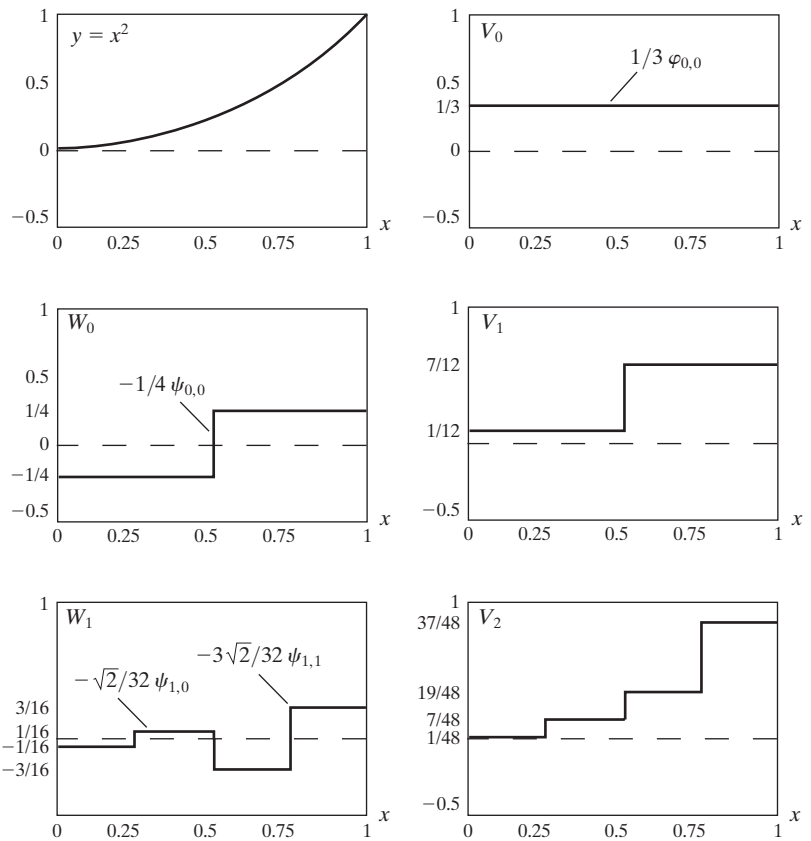



**FIGURE 7.15**
A wavelet series expansion of $y = x^2$ using Haar wavelets.

The first term in this expansion uses $c_0(0)$ to generate a subspace $V_0$ approximation of the function being expanded. This approximation is shown in Fig. 7.15(b) and is the average value of the original function. The second term uses $d_0(0)$ to refine the approximation by adding a level of detail from subspace $W_0$. The added detail and resulting $V_1$ approximation are shown in Figs. 7.15(c) and (d), respectively. Another level of detail is added by the subspace $W_1$ coefficients $d_1(0)$ and $d_1(1)$. This additional detail is shown in Fig. 7.15(e), and the resulting $V_2$ approximation is depicted in 7.15(f). Note that the expansion is now beginning to resemble the original function. As higher scales (greater levels of detail) are added, the approximation becomes a more precise representation of the function, realizing it in the limit as $j \rightarrow \infty$. ■

### 7.3.2 The Discrete Wavelet Transform

Like the Fourier series expansion, the wavelet series expansion of the previous section maps a function of a continuous variable into a sequence of coefficients. If the function being expanded is discrete (i.e., a sequence of numbers), the resulting coefficients are called the *discrete wavelet transform* (DWT). For example, if $f(n) = f(x_0 + n\,\Delta x)$ for some $x_0$, $\Delta x$, and $n = 0, 1, 2, \ldots, M - 1$,

the wavelet series expansion coefficients for $f(x)$ [defined by Eqs. (7.3-2) and (7.3-3)] become the *forward* DWT coefficients for sequence $f(n)$:

$$W_\varphi(j_0, k) = \frac{1}{\sqrt{M}} \sum_n f(n)\varphi_{j_0,k}(n) \tag{7.3-5}$$

$$W_\psi(j, k) = \frac{1}{\sqrt{M}} \sum_n f(n)\psi_{j,k}(n) \quad \text{for } j \geq j_0 \tag{7.3-6}$$

The $\varphi_{j_0,k}(n)$ and $\psi_{j,k}(n)$ in these equations are sampled versions of basis functions $\varphi_{j_0,k}(x)$ and $\psi_{j,k}(x)$. For example, $\varphi_{j_0,k}(n) = \varphi_{j_0,k}(x_s + n\Delta x_s)$ for some $x_s$, $\Delta x_s$, and $n = 0, 1, 2, \ldots, M - 1$. Thus, we employ $M$ equally spaced samples over the support of the basis functions (see Example 7.8 below). In accordance with Eq. (7.3-1), the complementary *inverse* DWT is

$$f(n) = \frac{1}{\sqrt{M}} \sum_k W_\varphi(j_0, k)\varphi_{j_0,k}(n) + \frac{1}{\sqrt{M}} \sum_{j=j_0}^{\infty} \sum_k W_\psi(j, k)\psi_{j,k}(n) \tag{7.3-7}$$

Normally, we let $j_0 = 0$ and select $M$ to be a power of 2 (i.e., $M = 2^J$) so that the summations in Eqs. (7.3-5) through (7.3-7) are performed over $n = 0, 1, 2, \ldots, M - 1$, $j = 0, 1, 2, \ldots, J - 1$, and $k = 0, 1, 2, \ldots, 2^j - 1$. For Haar wavelets, the discretized scaling and wavelet functions employed in the transform (i.e., the basis functions) correspond to the rows of the $M \times M$ Haar transformation matrix of Section 7.1.3. The transform itself is composed of $M$ coefficients, the minimum scale is 0, and the maximum scale is $J - 1$. For reasons noted in Section 7.3.1 and illustrated in Example 7.6, the coefficients defined in Eqs. (7.3-5) and (7.3-6) are usually called *approximation* and *detail coefficients*, respectively.

The $W_\varphi(j_0, k)$ and $W_\psi(j, k)$ in Eqs. (7.3-5) to (7.3-7) correspond to the $c_{j_0}(k)$ and $d_j(k)$ of the wavelet series expansion in the previous section. (This change of variables is not necessary but paves the way for the standard notation used for the continuous wavelet transform of the next section.) Note that the integrations in the series expansion have been replaced by summations, and a $1/\sqrt{M}$ normalizing factor, reminiscent of the DFT in Section 4.4.1, has been added to both the forward and inverse expressions. This factor alternately could be incorporated into the forward or inverse alone as $1/M$. Finally, it should be remembered that Eqs. (7.3-5) through (7.3-7) are valid for orthonormal bases and tight frames alone. For biorthogonal bases, the $\varphi$ and $\psi$ terms in Eqs. (7.3-5) and (7.3-6) must be replaced by their duals, $\tilde{\varphi}$ and $\tilde{\psi}$, respectively.

**EXAMPLE 7.8:** Computing a one-dimensional discrete wavelet transform.

■ To illustrate the use of Eqs. (7.3-5) through (7.3-7), consider the discrete function of four points: $f(0) = 1$, $f(1) = 4$, $f(2) = -3$, and $f(3) = 0$. Because $M = 4$, $J = 2$ and, with $j_0 = 0$, the summations are performed over $x = 0, 1, 2, 3$, $j = 0, 1$, and $k = 0$ for $j = 0$ or $k = 0, 1$ for $j = 1$. We will use the Haar scaling and wavelet functions and assume that the four samples of

$f(x)$ are distributed over the support of the basis functions, which is 1 in width. Substituting the four samples into Eq. (7.3-5), we find that

$$W_\varphi(0,0) = \frac{1}{2}\sum_{n=0}^{3} f(n)\varphi_{0,0}(n)$$
$$= \frac{1}{2}\left[1\cdot 1 + 4\cdot 1 - 3\cdot 1 + 0\cdot 1\right] = 1$$

because $\varphi_{0,0}(n) = 1$ for $n = 0, 1, 2, 3$. Note that we have employed uniformly spaced samples of the Haar scaling function for $j = 0$ and $k = 0$. The values correspond to the first row of Haar transformation matrix $\mathbf{H}_4$ of Section 7.1.3. Continuing with Eq. (7.3-6) and similarly spaced samples of $\psi_{j,k}(x)$, which correspond to rows 2, 3, and 4 of $\mathbf{H}_4$, we get

$$W_\psi(0,0) = \frac{1}{2}\left[1\cdot 1 + 4\cdot 1 - 3\cdot(-1) + 0\cdot(-1)\right] = 4$$

$$W_\psi(1,0) = \frac{1}{2}\left[1\cdot\sqrt{2} + 4\cdot\left(-\sqrt{2}\right) - 3\cdot 0 + 0\cdot 0\right] = -1.5\sqrt{2}$$

$$W_\psi(1,1) = \frac{1}{2}\left[1\cdot 0 + 4\cdot 0 - 3\cdot\sqrt{2} + 0\cdot\left(-\sqrt{2}\right)\right] = -1.5\sqrt{2}$$

Thus, the discrete wavelet transform of our simple four-sample function relative to the Haar wavelet and scaling function is $\left\{1, 4, -1.5\sqrt{2}, -1.5\sqrt{2}\right\}$, where the transform coefficients have been arranged in the order in which they were computed.

Equation (7.3-7) lets us reconstruct the original function from its transform. Iterating through its summation indices, we get

$$f(n) = \frac{1}{2}\left[W_\varphi(0,0)\,\varphi_{0,0}(n) + W_\psi(0,0)\psi_{0,0}(n) + W_\psi(1,0)\,\psi_{1,0}(n) + W_\psi(1,1)\psi_{1,1}(n)\right]$$

for $n = 0, 1, 2, 3$. If $n = 0$, for instance,

$$f(0) = \frac{1}{2}\left[1\cdot 1 + 4\cdot 1 - 1.5\sqrt{2}\cdot\left(\sqrt{2}\right) - 1.5\sqrt{2}\cdot 0\right] = 1$$

As in the forward case, uniformly spaced samples of the scaling and wavelet functions are used in the computation of the inverse. ■

The four-point DWT in the preceding example is an illustration of a two-scale decomposition of $f(n)$—that is, $j = \{0, 1\}$. The underlying assumption was that starting scale $j_0$ was zero, but other starting scales are possible. It is left as an exercise for the reader (see Problem 7.16) to compute the single-scale transform $\left\{2.5\sqrt{2}, -1.5\sqrt{2}, -1.5\sqrt{2}, -1.5\sqrt{2}\right\}$, which results when the starting scale is 1. Thus, Eqs. (7.3-5) and (7.3-6) define a "family" of transforms that differ in starting scale $j_0$.

### 7.3.3 The Continuous Wavelet Transform

The natural extension of the discrete wavelet transform is the *continuous wavelet transform* (CWT), which transforms a continuous function into a highly redundant function of two continuous variables—translation and scale. The resulting transform is easy to interpret and valuable for time-frequency analysis. Although our interest is in discrete images, the continuous transform is covered here for completeness.

The continuous wavelet transform of a continuous, square-integrable function, $f(x)$, relative to a real-valued wavelet, $\psi(x)$, is defined as

$$W_\psi(s, \tau) = \int_{-\infty}^{\infty} f(x)\psi_{s,\tau}(x)\, dx \tag{7.3-8}$$

where

$$\psi_{s,\tau}(x) = \frac{1}{\sqrt{s}}\psi\left(\frac{x - \tau}{s}\right) \tag{7.3-9}$$

and $s$ and $\tau$ are called *scale* and *translation* parameters, respectively. Given $W_\psi(s, \tau)$, $f(x)$ can be obtained using the *inverse continuous wavelet transform*

$$f(x) = \frac{1}{C_\psi}\int_0^{\infty}\int_{-\infty}^{\infty} W_\psi(s, \tau)\frac{\psi_{s,\tau}(x)}{s^2}d\tau\, ds \tag{7.3-10}$$

where

$$C_\psi = \int_{-\infty}^{\infty}\frac{|\Psi(\mu)|^2}{|\mu|}d\mu \tag{7.3-11}$$

and $\Psi(\mu)$ is the Fourier transform of $\psi(x)$. Equations (7.3-8) through (7.3-11) define a reversible transformation as long as the so-called *admissibility criterion*, $C_\psi < \infty$, is satisfied (Grossman and Morlet [1984]). In most cases, this simply means that $\Psi(0) = 0$ and $\Psi(\mu) \to 0$ as $\mu \to \infty$ fast enough to make $C_\psi < \infty$.

The preceding equations are reminiscent of their discrete counterparts—Eqs. (7.2-19), (7.3-1), (7.3-3), (7.3-6), and (7.3-7). The following similarities should be noted:

1. The continuous translation parameter, $\tau$, takes the place of the integer translation parameter, $k$.
2. The continuous scale parameter, $s$, is inversely related to the binary scale parameter, $2^j$. This is because $s$ appears in the denominator of $\psi\big((x - \tau)/s\big)$ in Eq. (7.3-9). Thus, wavelets used in continuous transforms are compressed or reduced in width when $0 < s < 1$ and dilated or expanded when $s > 1$. Wavelet scale and our traditional notion of frequency are inversely related.

3. The continuous transform is similar to a series expansion [see Eq. (7.3-1)] or discrete transform [see Eq. (7.3-6)] in which the starting scale $j_0 = -\infty$. This—in accordance with Eq. (7.2-26)—eliminates explicit scaling function dependence, so that the function is represented in terms of wavelets alone.
4. Like the discrete transform, the continuous transform can be viewed as a set of transform coefficients, $\{W_\psi(s, \tau)\}$, that measure the similarity of $f(x)$ with a set of basis functions, $\{\psi_{s,\tau}(x)\}$. In the continuous case, however, both sets are infinite. Because $\psi_{s,\tau}(x)$ is real valued and $\psi_{s,\tau}(x) = \psi^*_{s,\tau}(x)$, each coefficient from Eq. (7.3-8) is the integral inner product, $\langle f(x), \psi_{s,\tau}(x)\rangle$, of $f(x)$ and $\psi_{s,\tau}(x)$.

**EXAMPLE 7.9:** A one-dimensional continuous wavelet transform.

■ The *Mexican hat* wavelet,

$$\psi(x) = \left(\frac{2}{\sqrt{3}}\pi^{-1/4}\right)(1 - x^2)e^{-x^2/2} \tag{7.3-12}$$

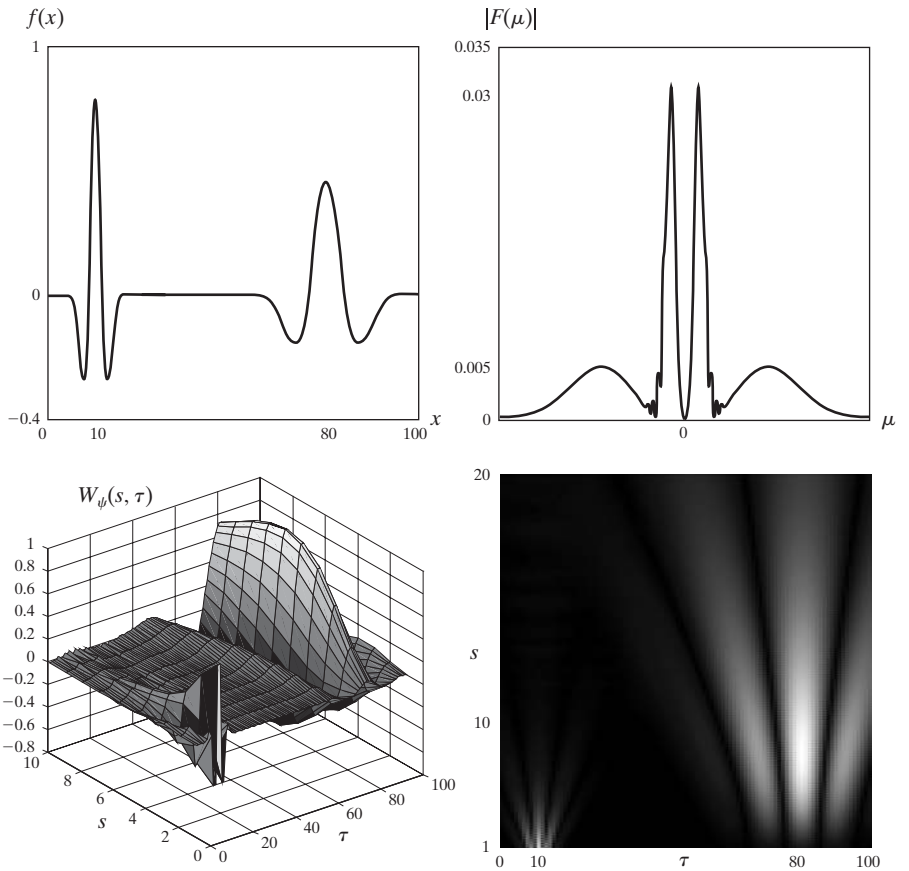
gets its name from its distinctive shape [see Fig. 7.16(a)]. It is proportional to the second derivative of the Gaussian probability function, has an average value of 0, and is compactly supported (i.e., dies out rapidly as $|x| \to \infty$). Although it satisfies the admissibility requirement for the existence of continuous, reversible transforms, there is not an associated scaling function, and the computed transform does not result in an orthogonal analysis. Its most distinguishing features are its symmetry and the existence of the explicit expression of Eq. (7.3-12).

The continuous, one-dimensional function in Fig. 7.16(a) is the sum of two Mexican hat wavelets:

$$f(x) = \psi_{1,10}(x) + \psi_{6,80}(x)$$

Its Fourier spectrum, shown in Fig. 7.16(b), reveals the close connection between scaled wavelets and Fourier frequency bands. The spectrum contains two broad frequency bands (or peaks) that correspond to the two Gaussian-like perturbations of the function.

Figure 7.16(c) shows a portion ($1 \le s \le 10$ and $\tau \le 100$) of the CWT of the function in Fig. 7.16(a) relative to the Mexican hat wavelet. Unlike the Fourier spectrum in Fig. 7.16(b), it provides both spatial and frequency information. Note, for example, that when $s = 1$, the transform achieves a maximum at $\tau = 10$, which corresponds to the location of the $\psi_{1,10}(x)$ component of $f(x)$. Because the transform provides an objective measure of the similarity between $f(x)$ and the wavelets for which it is computed, it is easy to see how it can be used for feature detection. We simply need wavelets that match the features of interest. Similar observations can be drawn from the intensity plot in Fig. 7.16(d), where the absolute value of the transform $|W_\psi(s, \tau)|$ is displayed as intensities between black and white. Note that the continuous wavelet transform turns a 1-D function into a 2-D result. ■




a b
c d

**FIGURE 7.16**
The continuous wavelet transform (c and d) and Fourier spectrum (b) of a continuous 1-D function (a).

## 7.4 The Fast Wavelet Transform

The *fast wavelet transform* (FWT) is a computationally efficient implementation of the discrete wavelet transform (DWT) that exploits a surprising but fortunate relationship between the coefficients of the DWT at adjacent scales. Also called *Mallat's herringbone algorithm* (Mallat [1989a, 1989b]), the FWT resembles the two-band subband coding scheme of Section 7.1.2.

Consider again the multiresolution refinement equation

$$\varphi(x) = \sum_n h_\varphi(n)\sqrt{2}\varphi(2x - n) \tag{7.4-1}$$

Equation (7.4-1) is Eq. (7.2-18) of Section 7.2.2.

Scaling $x$ by $2^j$, translating it by $k$, and letting $m = 2k + n$ gives

$$\begin{aligned}\varphi(2^j x - k) &= \sum_n h_\varphi(n)\sqrt{2}\varphi\left(2(2^j x - k) - n\right) \\ &= \sum_m h_\varphi(n)\sqrt{2}\varphi(2^{j+1}x - 2k - n) \\ &= \sum_m h_\varphi(m - 2k)\sqrt{2}\varphi(2^{j+1}x - m)\end{aligned} \tag{7.4-2}$$

Note that scaling vector $h_\varphi$ can be thought of as the "weights" used to expand $\varphi(2^j x - k)$ as a sum of scale $j + 1$ scaling functions. A similar sequence of operations—beginning with Eq. (7.2-28)—provides an analogous result for $\psi(2^j x - k)$. That is,

$$\psi(2^j x - k) = \sum_m h_\psi(m - 2k)\sqrt{2}\varphi(2^{j+1}x - m) \tag{7.4-3}$$

where scaling vector $h_\varphi(n)$ in Eq. (7.4-2) corresponds to wavelet vector $h_\psi(n)$ in Eq. (7.4-3).

The wavelet series expansion coefficients become the DWT coefficient when $f$ is discrete. Here, we begin with the series expansion coefficients to simplify the derivation; we will be able to substitute freely from earlier results (like the scaling and wavelet function definitions).

Now consider Eqs. (7.3-2) and (7.3-3) of Section 7.3.1. They define the wavelet series expansion coefficients of continuous function $f(x)$. Substituting Eq. (7.2-19)—the wavelet defining equation—into Eq. (7.3-3), we get

$$d_j(k) = \int f(x)2^{j/2}\psi(2^j x - k)\,dx \tag{7.4-4}$$

which, upon replacing $\psi(2^j x - k)$ with the right side of Eq. (7.4-3), becomes

$$d_j(k) = \int f(x)2^{j/2}\left[\sum_m h_\psi(m - 2k)\sqrt{2}\varphi(2^{j+1}x - m)\right]dx \tag{7.4-5}$$

Interchanging the sum and integral and rearranging terms then gives

$$d_j(k) = \sum_m h_\psi(m - 2k)\left[\int f(x)2^{(j+1)/2}\varphi(2^{j+1}x - m)\right] \tag{7.4-6}$$

where the bracketed quantity is $c_{j_0}(k)$ of Eq. (7.3-2) with $j_0 = j + 1$ and $k = m$. To see this, substitute Eq. (7.2-10) into Eq. (7.3-2) and replace $j_0$ and $k$ with $j + 1$ and $m$, respectively. Therefore, we can write

$$d_j(k) = \sum_m h_\psi(m - 2k)c_{j+1}(m) \tag{7.4-7}$$

and note that the detail coefficients at scale $j$ are a function of the approximation coefficients at scale $j + 1$. Using Eqs. (7.4-2) and (7.3-2) as the starting point of a similar derivation involving the wavelet series expansion (and DWT) approximation coefficients, we find similarly that

$$c_j(k) = \sum_m h_\varphi(m - 2k)c_{j+1}(m) \tag{7.4-8}$$

Because the $c_j(k)$ and $d_j(k)$ coefficients of the wavelet series expansion become the $W_\varphi(j, k)$ and $W_\psi(j, k)$ coefficients of the DWT when $f(x)$ is discrete (see Section 7.3.2), we can write

$$W_\psi(j, k) = \sum_m h_\psi(m - 2k) W_\varphi(j + 1, m) \tag{7.4-9}$$

$$W_\varphi(j, k) = \sum_m h_\varphi(m - 2k) W_\varphi(j + 1, m) \tag{7.4-10}$$

Equations (7.4-9) and (7.4-10) reveal a remarkable relationship between the DWT coefficients of adjacent scales. Comparing these results to Eq. (7.1-7), we see that both $W_\varphi(j, k)$ and $W_\psi(j, k)$, the scale $j$ approximation and the detail coefficients, can be computed by convolving $W_\varphi(j + 1, k)$, the scale $j + 1$ approximation coefficients, with the order-reversed scaling and wavelet vectors, $h_\varphi(-n)$ and $h_\psi(-n)$, and subsampling the results. Figure 7.17 summarizes these operations in block diagram form. Note that this diagram is identical to the analysis portion of the two-band subband coding and decoding system of Fig. 7.6, with $h_0(n) = h_\varphi(-n)$ and $h_1(n) = h_\psi(-n)$. Therefore, we can write

$$W_\psi(j, k) = h_\psi(-n) \star W_\varphi(j + 1, n)\Big|_{n=2k, k\geq 0} \tag{7.4-11}$$

and

$$W_\varphi(j, k) = h_\varphi(-n) \star W_\varphi(j + 1, n)\Big|_{n=2k, k\geq 0} \tag{7.4-12}$$

where the convolutions are evaluated at instants $n = 2k$ for $k \geq 0$. As will be shown in Example 7.10, evaluating convolutions at nonnegative, even indices is equivalent to filtering and downsampling by 2.

If $h_\varphi(m - 2k)$ in Eq. (7.4-9) is rewritten as $h_\varphi(-(2k - m))$, we see that the first minus sign is responsible for the order reversal [see Eq. (7.1-6)], the $2k$ is responsible for the subsampling [see Eq. (7.1-2)], and $m$ is the dummy variable for convolution [see Eq. (7.1-7)].

Equations (7.4-11) and (7.4-12) are the defining equations for the computation of the fast wavelet transform. For a sequence of length $M = 2^J$, the number of mathematical operations involved is on the order of $O(M)$. That is, the number of multiplications and additions is linear with respect to the length of the input sequence—because the number of multiplications and additions involved in the convolutions performed by the FWT analysis bank in Fig. 7.17 is proportional to the length of the sequences being convolved. Thus, the FWT compares favorably with the FFT algorithm, which requires on the order of $O(M \log_2 M)$ operations.

To conclude the development of the FWT, we simply note that the filter bank in Fig. 7.17 can be "iterated" to create multistage structures for computing DWT coefficients at two or more successive scales. For example, Fig. 7.18(a) shows a two-stage filter bank for generating the coefficients at the two highest scales of the transform. Note that the highest scale coefficients are assumed to be samples of the function itself. That is, $W_\varphi(J, n) = f(n)$, where $J$ is the highest

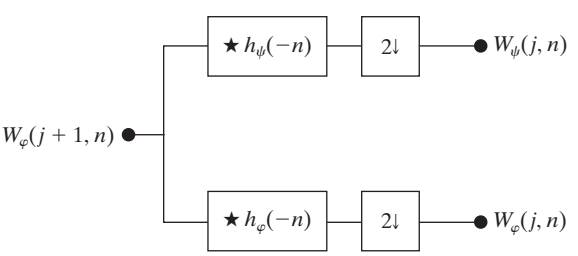



**FIGURE 7.17** An FWT analysis bank.

a
b

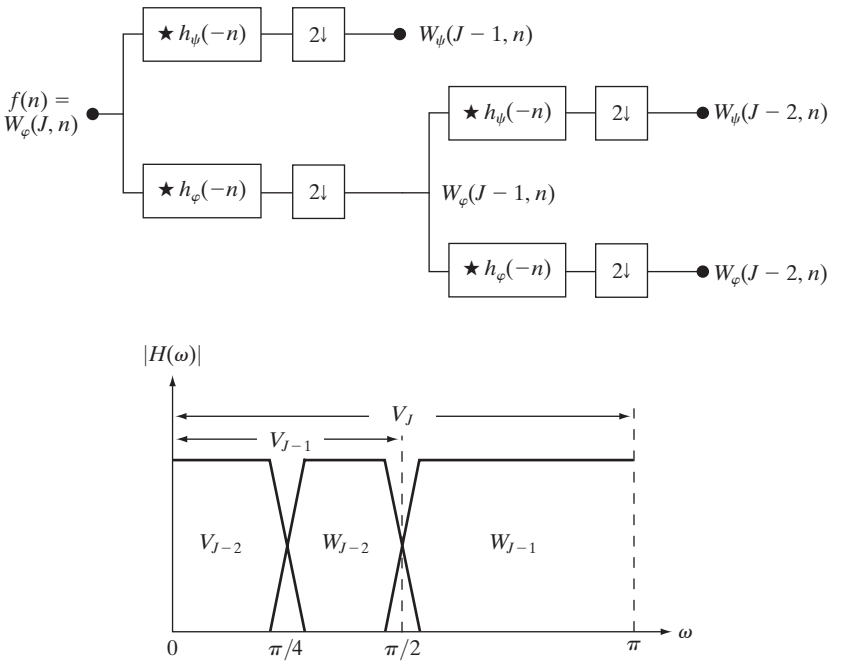



**FIGURE 7.18**
(a) A two-stage or two-scale FWT analysis bank and (b) its frequency splitting characteristics.

scale. [In accordance with Section 7.2.2, $f(x) \in V_J$, where $V_J$ is the scaling space in which $f(x)$ resides.] The first filter bank in Fig. 7.18(a) splits the original function into a lowpass, approximation component, which corresponds to scaling coefficients $W_\varphi(J-1, n)$; and a highpass, detail component, corresponding to coefficients $W_\psi(J-1, n)$. This is illustrated graphically in Fig. 7.18(b), where scaling space $V_J$ is split into wavelet subspace $W_{J-1}$ and scaling subspace $V_{J-1}$. The spectrum of the original function is split into two half-band components. The second filter bank of Fig. 7.18(a) splits the spectrum and subspace $V_{J-1}$, the lower half-band, into quarter-band subspaces $W_{J-2}$ and $V_{J-2}$ with corresponding DWT coefficients $W_\psi(J-2, n)$ and $W_\varphi(J-2, n)$, respectively.

The two-stage filter bank of Fig. 7.18(a) is extended easily to any number of scales. A third filter bank, for example, would operate on the $W_\varphi(J-2, n)$ coefficients, splitting scaling space $V_{J-2}$ into two eighth-band subspaces $W_{J-3}$ and $V_{J-3}$. Normally, we choose $2^J$ samples of $f(x)$ and employ $P$ filter banks (as in Fig. 7.17) to generate a $P$-scale FWT at scales $J-1, J-2, \ldots, J-P$. The highest scale (i.e., $J-1$) coefficients are computed first; the lowest scale (i.e., $J-P$) last. If function $f(x)$ is sampled above the Nyquist rate, as is usually the case, its samples are good approximations of the scaling coefficients at the sampling resolution and can be used as the starting high-resolution scaling coefficient inputs. In other words, no wavelet or detail coefficients are needed at the sampling scale. The highest-resolution scaling functions act as unit discrete impulse functions in Eqs. (7.3-5) and (7.3-6), allowing $f(n)$ to be used as the scaling (approximation) input to the first two-band filter bank (Odegard, Gopinath, and Burrus [1992]).

**EXAMPLE 7.10:** Computing a 1-D fast wavelet transform.

■ To illustrate the preceding concepts, consider the discrete function $f(n) = \{1, 4, -3, 0\}$ from Example 7.8. As in that example, we will compute the transform based on Haar scaling and wavelet functions. Here, however, we will not use the basis functions directly, as was done in the DWT of Example 7.8. Instead, we will use the corresponding scaling and wavelet vectors from Examples 7.5 and 7.6:

$$h_\varphi(n) = \begin{cases} 1/\sqrt{2} & n = 0, 1 \\ 0 & \text{otherwise} \end{cases} \tag{7.4-13}$$

and

$$h_\psi(n) = \begin{cases} 1/\sqrt{2} & n = 0 \\ -1/\sqrt{2} & n = 1 \\ 0 & \text{otherwise} \end{cases} \tag{7.4-14}$$

These are the functions used to build the FWT filter banks; they provide the filter coefficients. Note that because Haar scaling and wavelet functions are orthonormal, Eq. (7.1-14) can be used to generate the FWT filter coefficients from a single prototype filter—like $h_\varphi(n)$ in Table 7.2, which corresponds to $g_0(n)$ in Eq. (7.1-14):

Since the DWT computed in Example 7.8 was composed of elements $\{W_\varphi(0, 0), W_\psi(0, 0), W_\psi(1, 0), W_\psi(1, 1)\}$, we will compute the corresponding two-scale FWT for scales $j = \{0, 1\}$. That is, $J = 2$ (there are $2^J = 2^2$ samples) and $P = 2$ (we are working with scales $J - 1 = 2 - 1 = 1$ and $J - P = 2 - 2 = 0$ in that order). The transform will be computed using the two-stage filter bank of Fig. 7.18(a). Figure 7.19 shows the sequences that result from the required FWT convolutions and downsamplings. Note that function $f(n)$ itself is the scaling (approximation) input to the leftmost filter bank. To compute the $W_\psi(1, k)$ coefficients that appear at the end of the upper branch of Fig. 7.19, for example, we first convolve $f(n)$ with $h_\psi(-n)$. As explained in Section 3.4.2, this requires flipping one of the functions about the origin, sliding it past the other, and computing the sum of the point-wise product of the two functions. For sequences $\{1, 4, -3, 0\}$ and $\{-1/\sqrt{2}, 1/\sqrt{2}\}$, this produces

$$\{-1/\sqrt{2}, -3/\sqrt{2}, 7/\sqrt{2}, -3/\sqrt{2}, 0\}$$

where the second term corresponds to index $k = 2n = 0$. (In Fig. 7.19, underlined values represent negative indices, i.e., $n < 0$.) When downsampled by

| $n$ | $h_\varphi(n)$ |
|---|---|
| 0 | $1/\sqrt{2}$ |
| 1 | $1/\sqrt{2}$ |

**TABLE 7.2** Orthonormal Haar filter coefficients for $h_\varphi(n)$.

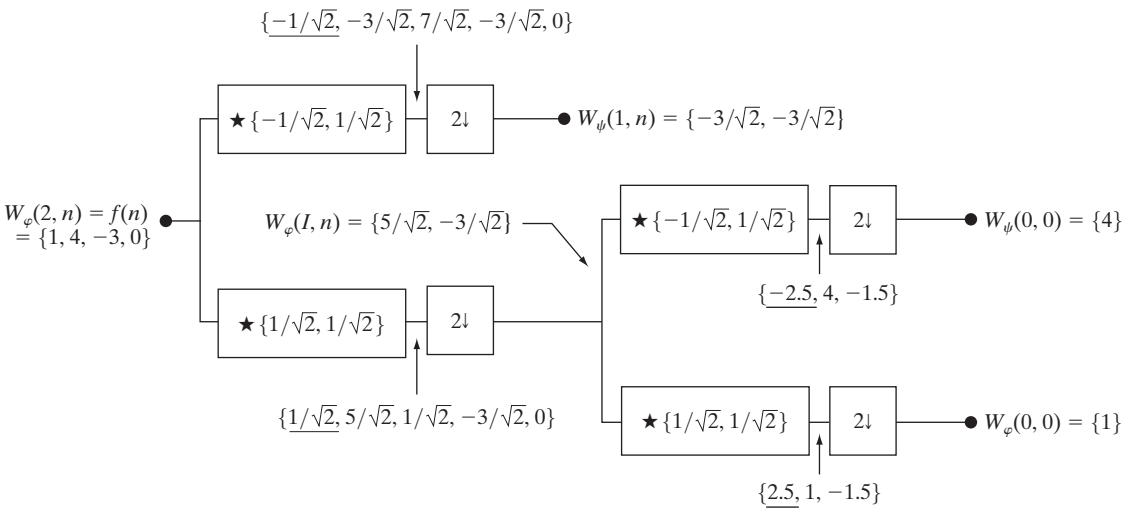



**FIGURE 7.19** Computing a two-scale fast wavelet transform of sequence $\{1, 4, -3, 0\}$ using Haar scaling and wavelet vectors.

taking the even-indexed points, we get $W_\psi(1, k) = \left\{-3/\sqrt{2}, -3/\sqrt{2}\right\}$ for $k = \{0, 1\}$. Alternatively, we can use Eq. (7.4-12) to compute

$$W_\psi(1, k) = h_\psi(-n) \star W_\varphi(2, n)\Big|_{n=2k, k\geq 0} = h_\psi(-n) \star f(n)\Big|_{n=2k, k\geq 0}$$

$$= \sum_l h_\psi(l - 2k)x(l)\Big|_{k=0,1}$$

$$= \frac{1}{\sqrt{2}}\, x(2k) - \frac{1}{\sqrt{2}}\, x(2k + 1)\Big|_{k=0,1}$$

Here, we have substituted $2k$ for $n$ in the convolution and employed $l$ as a dummy variable of convolution (i.e., for displacing the two sequences relative to one another). There are only two terms in the expanded sum because there are only two nonzero values in the order-reversed wavelet vector $h_\psi(-n)$. Substituting $k = 0$, we find that $W_\psi(1, 0) = -3/\sqrt{2}$; for $k = 1$, we get $W_\psi(1, 1) = -3/\sqrt{2}$. Thus, the filtered and downsampled sequence is $\left\{-3/\sqrt{2}, -3/\sqrt{2}\right\}$, which matches the earlier result. The remaining convolutions and downsamplings are performed in a similar manner. ■

As one might expect, a fast inverse transform for the reconstruction of $f(n)$ from the results of the forward transform can be formulated. Called the *inverse fast wavelet transform* ($\text{FWT}^{-1}$), it uses the scaling and wavelet vectors employed in the forward transform, together with the level $j$ approximation

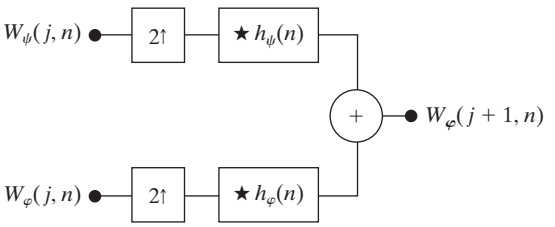



**FIGURE 7.20** The $\text{FWT}^{-1}$ synthesis filter bank.

and detail coefficients, to generate the level $j + 1$ approximation coefficients. Noting the similarity between the FWT analysis bank in Fig. 7.17 and the two-band subband analysis portion of Fig. 7.6(a), we can immediately postulate the required $\text{FWT}^{-1}$ *synthesis filter bank*. Figure 7.20 details its structure, which is identical to the synthesis portion of the two-band subband coding and decoding system in Fig. 7.6(a). Equation (7.1-14) of Section 7.1.2 defines the relevant synthesis filters. As noted there, perfect reconstruction (for two-band orthonormal filters) requires $g_i(n) = h_i(-n)$ for $i = \{0, 1\}$. That is, the synthesis and analysis filters must be order-reversed versions of one another. Since the FWT analysis filters (see Fig. 7.17) are $h_0(n) = h_\varphi(-n)$ and $h_1(n) = h_\psi(-n)$, the required $\text{FWT}^{-1}$ synthesis filters are $g_0(n) = h_0(-n) = h_\varphi(n)$ and $g_1(n) = h_1(-n) = h_\psi(n)$. It should be remembered, however, that it is possible also to use biorthogonal analysis and synthesis filters, which are not order-reversed versions of one another. Biorthogonal analysis and synthesis filters are cross-modulated per Eqs. (7.1-10) and (7.1-11).

The $\text{FWT}^{-1}$ filter bank in Fig. 7.20 implements the computation

$$W_\varphi(j+1, k) = h_\varphi(k) \star W_\varphi^{2\uparrow}(j, k) + h_\psi(k) \star W_\psi^{2\uparrow}(j, k) \Big|_{k \geq 0} \tag{7.4-15}$$

where $W^{2\uparrow}$ signifies upsampling by 2 [i.e., inserting zeros in $W$ as defined by Eq. (7.1-1) so that it is twice its original length]. The upsampled coefficients are filtered by convolution with $h_\varphi(n)$ and $h_\psi(n)$ and added to generate a higher scale approximation. In essence, a better approximation of sequence $f(n)$ with greater detail and resolution is created. As with the forward FWT, the inverse filter bank can be iterated as shown in Fig. 7.21, where a two-scale structure for computing the final two scales of a $\text{FWT}^{-1}$ reconstruction is depicted. This coefficient combining process can be extended to any number of scales and guarantees perfect reconstruction of sequence $f(n)$.

Remember that like in pyramid coding (see Section 7.1.1), wavelet transforms can be computed at a user-specified number of scales. For a $2^J \times 2^J$ image, for example, there are $1 + \log_2 J$ possible scales.

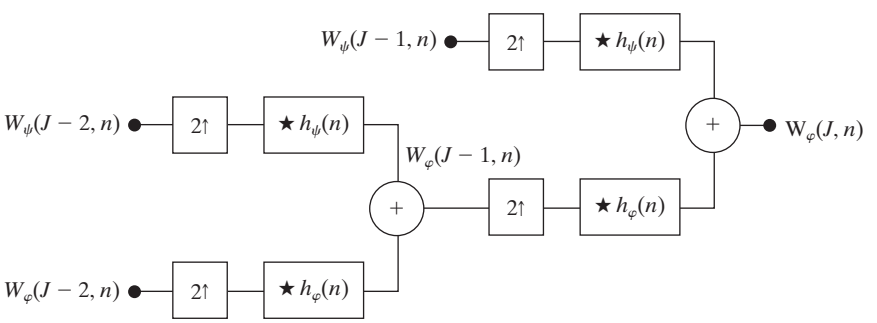



**FIGURE 7.21** A two-stage or two-scale $\text{FWT}^{-1}$ synthesis bank.

**EXAMPLE 7.11:** Computing a 1-D inverse fast wavelet transform.

■ Computation of the inverse fast wavelet transform mirrors its forward counterpart. Figure 7.22 illustrates the process for the sequence considered in Example 7.10. To begin the calculation, the level 0 approximation and detail coefficients are upsampled to yield $\{1, 0\}$ and $\{4, 0\}$, respectively. Convolution with filters $g_0(n) = h_\varphi(n) = \left\{1/\sqrt{2}, 1/\sqrt{2}\right\}$ and $g_1(n) = h_\psi(n) = \left\{1/\sqrt{2}, -1/\sqrt{2}\right\}$ produces $\left\{1/\sqrt{2}, 1/\sqrt{2}, 0\right\}$ and $\left\{4/\sqrt{2}, -4/\sqrt{2}, 0\right\}$, which when added give $W_\varphi(1, n) = \left\{5/\sqrt{2}, -3/\sqrt{2}\right\}$. Thus, the level 1 approximation of Fig. 7.22, which matches the computed approximation in Fig. 7.19, is reconstructed. Continuing in this manner, $f(n)$ is formed at the right of the second synthesis filter bank. ■

We conclude our discussion of the fast wavelet transform by noting that while the Fourier basis functions (i.e., sinusoids) guarantee the existence of the FFT, the existence of the FWT depends upon the availability of a scaling function for the wavelets being used, as well as the orthogonality (or biorthogonality) of the scaling function and corresponding wavelets. Thus, the Mexican hat wavelet of Eq. (7.3-12), which does not have a companion scaling function, cannot be used in the computation of the FWT. In other words, we cannot construct a filter bank like that of Fig. 7.17 for the Mexican hat wavelet; it does not satisfy the underlying assumptions of the FWT approach.

Finally, we note that while time and frequency usually are viewed as different domains when representing functions, they are inextricably linked. When you try to analyze a function simultaneously in time and frequency, you run into the following problem: If you want precise information about time, you must accept some vagueness about frequency, and vice versa. This is the *Heisenberg uncertainty principle* applied to information processing. To illustrate the principle graphically, each basis function used in the representation of a function can be viewed schematically as a *tile* in a *time-frequency plane*. The tile, also called a *Heisenberg cell* or *Heisenberg box*, shows the frequency content of the basis function that it represents and where the basis function resides in time. Basis functions that are orthonormal are characterized by nonoverlapping tiles.

Figure 7.23 shows the time-frequency tiles for (a) an impulse function (i.e., conventional time domain) basis, (b) a sinusoidal (FFT) basis, and (c) an FWT

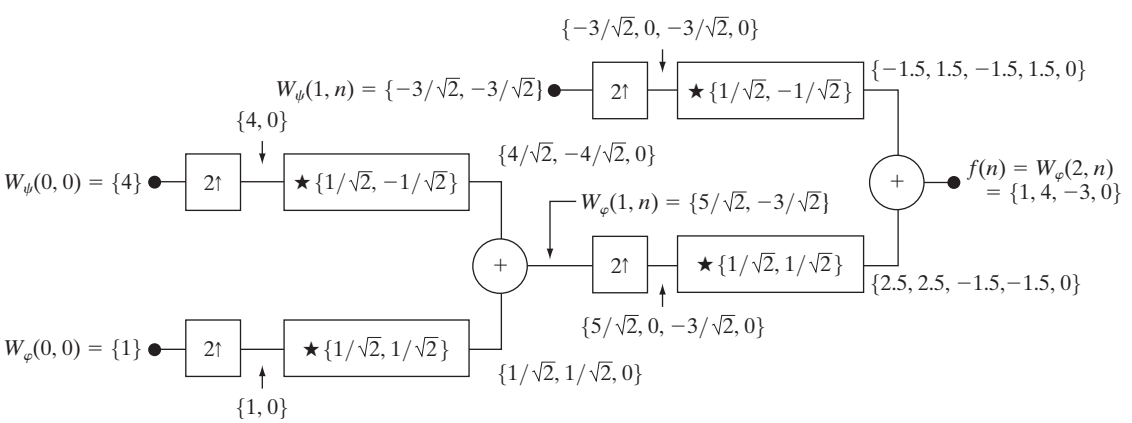
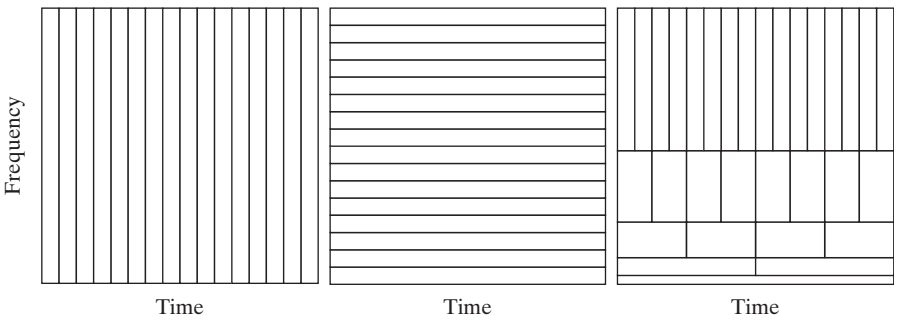



**FIGURE 7.22** Computing a two-scale inverse fast wavelet transform of sequence $\left\{1, 4, -1.5\sqrt{2}, -1.5\sqrt{2}\right\}$ with Haar scaling and wavelet functions.




a b c

**FIGURE 7.23** Time-frequency tilings for the basis functions associated with (a) sampled data, (b) the FFT, and (c) the FWT. Note that the horizontal strips of equal height rectangles in (c) represent FWT scales.

basis. Each tile is a rectangular region in Figs. 7.23(a) through (c); the height and width of the region defines the frequency and time characteristics of the functions that can be represented using the basis function. Note that the standard time domain basis in Fig. 7.23(a) pinpoints the instants when events occur but provides no frequency information [the width of each rectangle in Fig. 7.23(a) should be considered one instant in time]. Thus, to represent a single frequency sinusoid as an expansion using impulse basis functions, every basis function is required. The sinusoidal basis in Fig. 7.23(b), on the other hand, pinpoints the frequencies that are present in events that occur over long periods but provides no time resolution [the height of each rectangle in Fig. 7.23(b) should be considered a single frequency]. Thus, the single frequency sinusoid that was represented by an infinite number of impulse basis functions can be represented as an expansion involving one sinusoidal basis function. The time and frequency resolution of the FWT tiles in Fig. 7.23(c) vary, but the area of each tile (rectangle) is the same. At low frequencies, the tiles are shorter (i.e., have better frequency resolution or less ambiguity regarding frequency) but are wider (which corresponds to poorer time resolution or more ambiguity regarding time). At high frequencies, tile width is smaller (so the time resolution is improved) and tile height is greater (which means the frequency resolution is poorer). Thus, the FWT basis functions provide a compromise between the two limiting cases in Fig. 7.23(a) and (b). This fundamental difference between the FFT and FWT was noted in the introduction to the chapter and is important in the analysis of nonstationary functions whose frequencies vary in time.

## 7.5 Wavelet Transforms in Two Dimensions

The one-dimensional transforms of the previous sections are easily extended to two-dimensional functions like images. In two dimensions, a two-dimensional scaling function, $\varphi(x, y)$, and three two-dimensional wavelets, $\psi^H(x, y)$, $\psi^V(x, y)$, and $\psi^D(x, y)$, are required. Each is the product of two one-dimensional functions. Excluding products that produce one-dimensional results, like $\varphi(x)\psi(x)$, the four remaining products produce the *separable* scaling function

$$\varphi(x, y) = \varphi(x)\varphi(y) \tag{7.5-1}$$

and separable, "directionally sensitive" wavelets

$$\psi^H(x, y) = \psi(x)\varphi(y) \tag{7.5-2}$$

$$\psi^V(x, y) = \varphi(x)\psi(y) \tag{7.5-3}$$

$$\psi^D(x, y) = \psi(x)\psi(y) \tag{7.5-4}$$

These wavelets measure functional variations—intensity variations for images—along different directions: $\psi^H$ measures variations along columns (for example, horizontal edges), $\psi^V$ responds to variations along rows (like vertical edges), and $\psi^D$ corresponds to variations along diagonals. The directional sensitivity is a natural consequence of the separability in Eqs. (7.5-2) to (7.5-4); it does not increase the computational complexity of the 2-D transform discussed in this section.

Given separable two-dimensional scaling and wavelet functions, extension of the 1-D DWT to two dimensions is straightforward. We first define the scaled and translated basis functions:

$$\varphi_{j,m,n}(x, y) = 2^{j/2}\varphi(2^j x - m, 2^j y - n) \tag{7.5-5}$$

$$\psi^i_{j,m,n}(x, y) = 2^{j/2}\psi^i(2^j x - m, 2^j y - n), \quad i = \{H, V, D\} \tag{7.5-6}$$

where index $i$ identifies the directional wavelets in Eqs. (7.5-2) to (7.5-4). Rather than an exponent, $i$ is a superscript that assumes the values $H$, $V$, and $D$. The discrete wavelet transform of image $f(x, y)$ of size $M \times N$ is then

Now that we are dealing with 2-D images, $f(x, y)$ is a discrete function or sequence of values and $x$ and $y$ are discrete variables. The scaling and wavelet functions in Eq. (7.5-7) and (7.5-8) are sampled over their support (as was done in the 1-D case in Section 7.3.2).

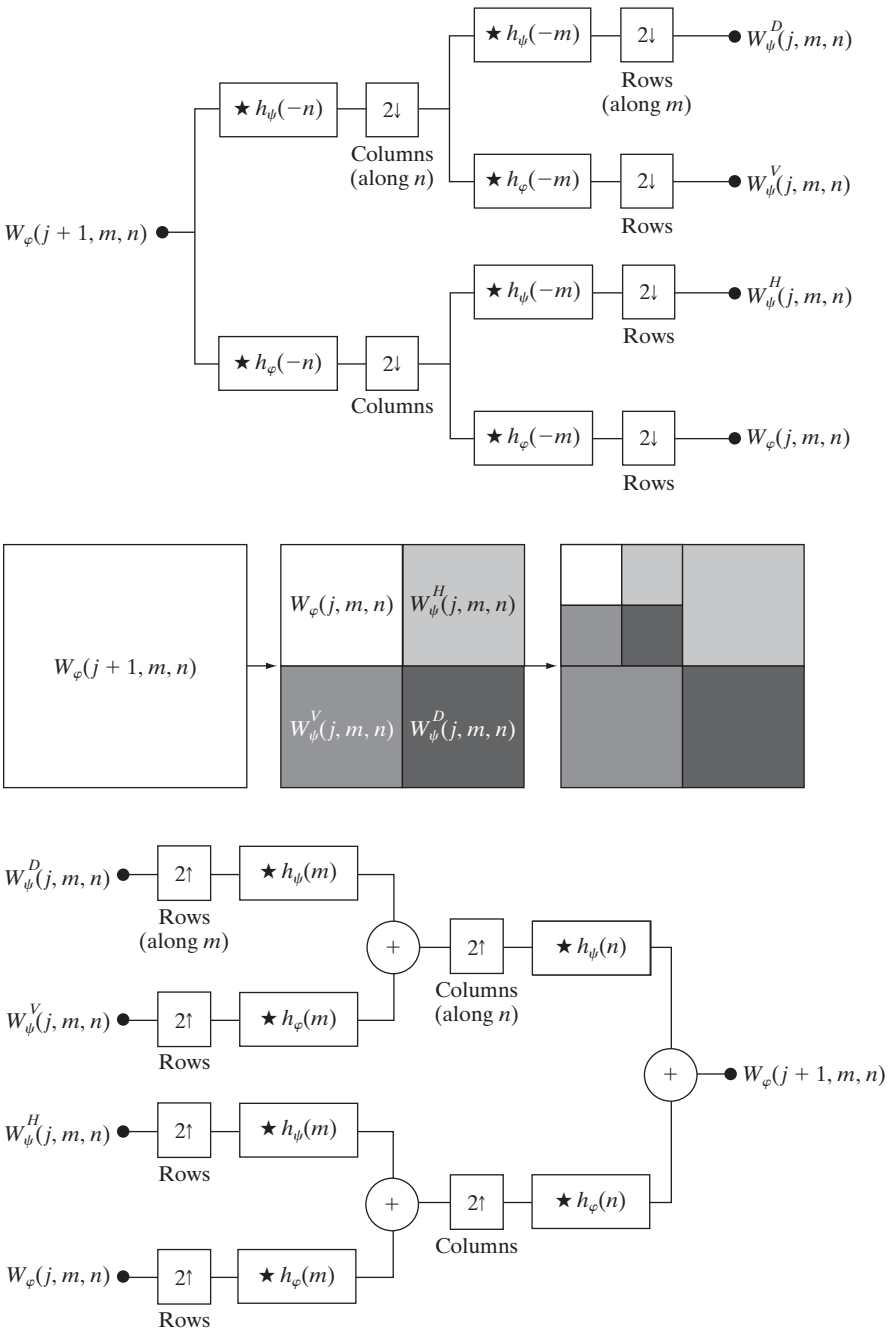
$$W_\varphi(j_0, m, n) = \frac{1}{\sqrt{MN}} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y)\varphi_{j_0,m,n}(x, y) \tag{7.5-7}$$

$$W^i_\psi(j, m, n) = \frac{1}{\sqrt{MN}} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y)\psi^i_{j,m,n}(x, y), \quad i = \{H, V, D\} \tag{7.5-8}$$

As in the one-dimensional case, $j_0$ is an arbitrary starting scale and the $W_\varphi(j_0, m, n)$ coefficients define an approximation of $f(x, y)$ at scale $j_0$. The $W^i_\psi(j, m, n)$ coefficients add horizontal, vertical, and diagonal details for scales $j \geq j_0$. We normally let $j_0 = 0$ and select $N = M = 2^J$ so that $j = 0, 1, 2, \ldots, J - 1$ and $m = n = 0, 1, 2, \ldots, 2^j - 1$. Given the $W_\varphi$ and $W^i_\psi$ of Eqs. (7.5-7) and (7.5-8), $f(x, y)$ is obtained via the inverse discrete wavelet transform

$$f(x, y) = \frac{1}{\sqrt{MN}} \sum_m \sum_n W_\varphi(j_0, m, n)\varphi_{j_0,m,n}(x, y) + \frac{1}{\sqrt{MN}} \sum_{i=H,V,D} \sum_{j=j_0}^{\infty} \sum_m \sum_n W^i_\psi(j, m, n)\psi^i_{j,m,n}(x, y) \tag{7.5-9}$$

Like the 1-D discrete wavelet transform, the 2-D DWT can be implemented using digital filters and downsamplers. With separable two-dimensional scaling and wavelet functions, we simply take the 1-D FWT of the rows of $f(x, y)$, followed by the 1-D FWT of the resulting columns. Figure 7.24(a) shows the




a
b
c

**FIGURE 7.24** The 2-D fast wavelet transform: (a) the analysis filter bank; (b) the resulting decomposition; and (c) the synthesis filter bank.

process in block diagram form. Note that, like its one-dimensional counterpart in Fig. 7.17, the 2-D FWT "filters" the scale $j + 1$ approximation coefficients to construct the scale $j$ approximation and detail coefficients. In the two-dimensional case, however, we get three sets of detail coefficients—the horizontal, vertical, and diagonal details.

The single-scale filter bank of Fig. 7.24(a) can be "iterated" (by tying the approximation output to the input of another filter bank) to produce a $P$ scale transform in which scale $j$ is equal to $J - 1, J - 2, \ldots, J - P$. As in the one-dimensional case, image $f(x, y)$ is used as the $W_\varphi(J, m, n)$ input. Convolving its rows with $h_\varphi(-n)$ and $h_\psi(-n)$ and downsampling its columns, we get two subimages whose horizontal resolutions are reduced by a factor of 2. The high-pass or detail component characterizes the image's high-frequency information with vertical orientation; the lowpass, approximation component contains its low-frequency, vertical information. Both subimages are then filtered columnwise and downsampled to yield four quarter-size output subimages—$W_\varphi$, $W_\psi^H$, $W_\psi^V$, and $W_\psi^D$. These subimages, which are shown in the middle of Fig. 7.24(b), are the inner products of $f(x, y)$ and the two-dimensional scaling and wavelet functions in Eqs. (7.5-1) through (7.5-4), followed by downsampling by two in each dimension. Two iterations of the filtering process produces the two-scale decomposition at the far right of Fig. 7.24(b).

Note how $W_\varphi$, $W_\psi^H$, $W_\psi^V$, and $W_\psi^D$ are arranged in Fig. 7.24(b). For each scale that is computed, they replace the previous scale approximation on which they were based.

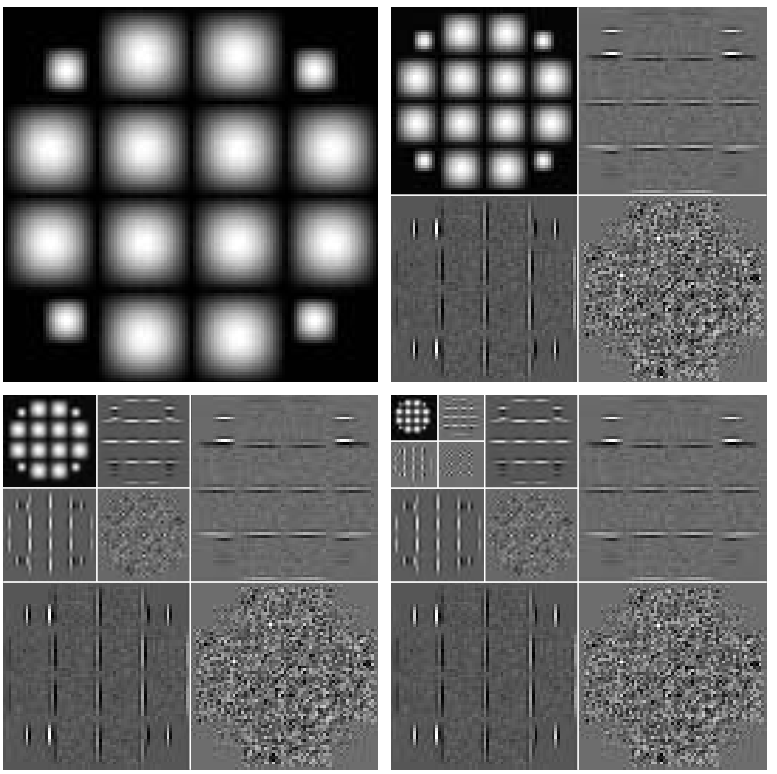
Figure 7.24(c) shows the synthesis filter bank that reverses the process just described. As would be expected, the reconstruction algorithm is similar to the one-dimensional case. At each iteration, four scale $j$ approximation and detail subimages are upsampled and convolved with two one-dimensional filters—one operating on the subimages' columns and the other on its rows. Addition of the results yields the scale $j + 1$ approximation, and the process is repeated until the original image is reconstructed.

**EXAMPLE 7.12:** Computing a 2-D fast wavelet transform.

■ Figure 7.25(a) is a $128 \times 128$ computer-generated image consisting of 2-D sine-like pulses on a black background. The objective of this example is to illustrate the mechanics involved in computing the 2-D FWT of this image. Figures 7.25(b) through (d) show three FWTs of the image in Fig. 7.25(a). The 2-D filter bank of Fig. 7.24(a) and the decomposition filters shown in Figs. 7.26(a) and (b) were used to generate all three results.

The scaling and wavelet vectors used in this example are described later. Our focus here is on the mechanics of the transform computation, which are independent of the filter coefficients employed.

Figure 7.25(b) shows the one-scale FWT of the image in Fig. 7.25(a). To compute this transform, the original image was used as the input to the filter bank of Fig. 7.24(a). The four resulting quarter-size decomposition outputs (i.e., the approximation and horizontal, vertical, and diagonal details) were then arranged in accordance with Fig. 7.24(b) to produce the image in Fig. 7.25(b). A similar process was used to generate the two-scale FWT in Fig. 7.25(c), but the input to the filter bank was changed to the quarter-size approximation subimage from the upper-left-hand corner of Fig. 7.25(b). As can be seen in Fig. 7.25(c), that quarter-size subimage was then replaced by the four quarter-size



a b
c d

**FIGURE 7.25**
Computing a 2-D three-scale FWT: (a) the original image; (b) a one-scale FWT; (c) a two-scale FWT; and (d) a three-scale FWT.

(now 1/16th of the size of the original image) decomposition results that were generated in the second filtering pass. Finally, Fig. 7.25(d) is the three-scale FWT that resulted when the subimage from the upper-left-hand corner of Fig. 7.25(c) was used as the filter bank input. Each pass through the filter bank produced four quarter-size output images that were substituted for the input from which they were derived. Note the directional nature of the wavelet-based subimages, $W_\psi^H$, $W_\psi^V$, and $W_\psi^D$, at each scale. ■

The decomposition filters used in the preceding example are part of a well-known family of wavelets called *symlets*, short for "symmetrical wavelets." Although they are not perfectly symmetrical, they are designed to have the least asymmetry and highest number of vanishing moments[†] for a given compact support (Daubechies [1992]). Figures 7.26(e) and (f) show the fourth-order

Recall that the compact support of a function is the interval in which the function has non-zero values.

[†]The $k$th moment of wavelet $\psi(x)$ is $m(k) = \int x^k \psi(x)\, dx$. Zero moments impact the smoothness of the scaling and wavelet functions and our ability to represent them as polynomials. An order-$N$ symlet has $N$ vanishing moments.

a b
c d
e f
g

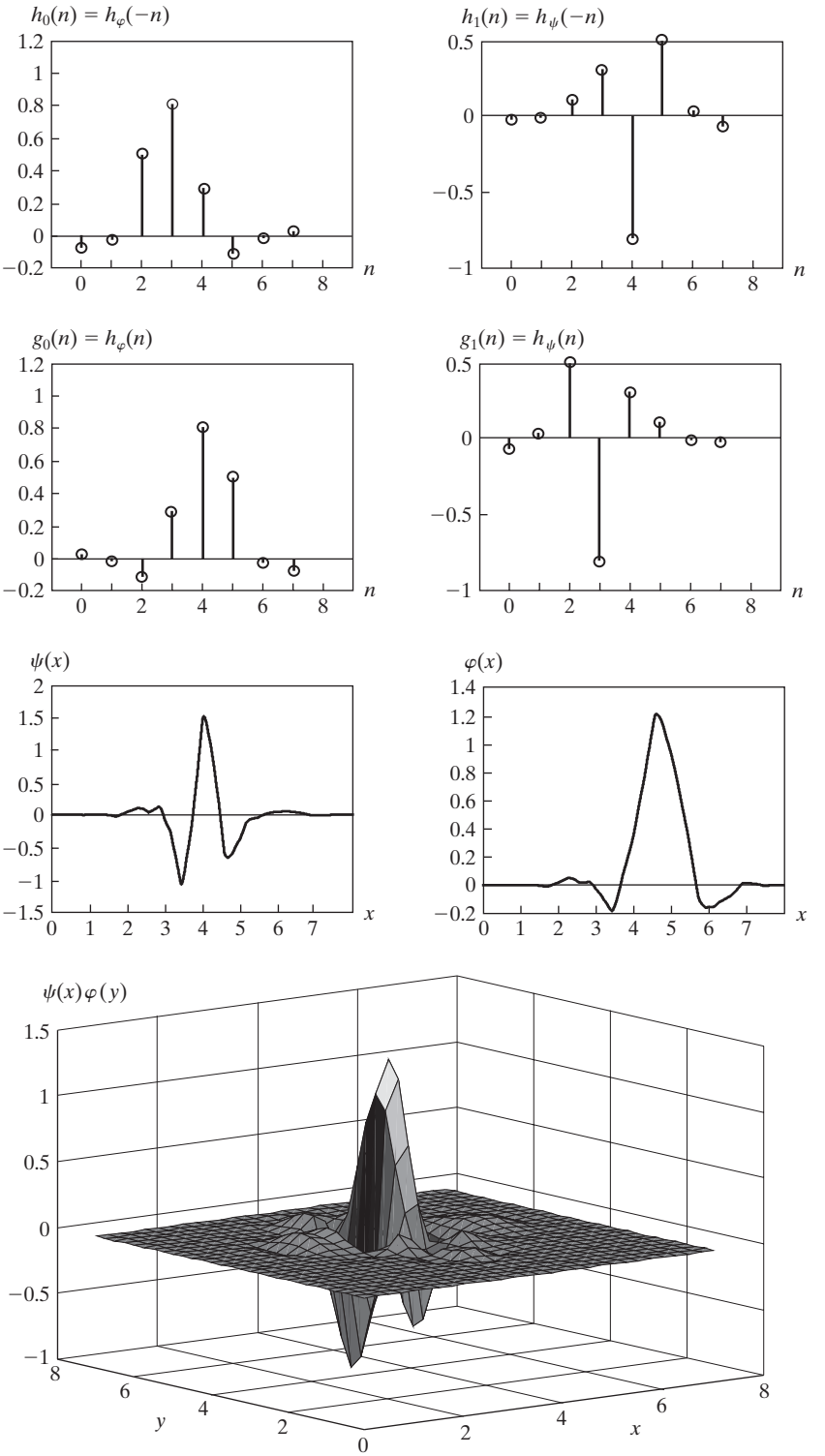



**FIGURE 7.26** Fourth-order symlets: (a)–(b) decomposition filters; (c)–(d) reconstruction filters; (e) the one-dimensional wavelet; (f) the one-dimensional scaling function; and (g) one of three two-dimensional wavelets, $\psi^V(x, y)$. See Table 7.3 for the values of $h_\varphi(n)$ for $0 \leq n \leq 7$.

| $n$ | $h_\varphi(n)$ |
|---|---|
| 0 | 0.0322 |
| 1 | −0.0126 |
| 2 | −0.0992 |
| 3 | 0.2979 |
| 4 | 0.8037 |
| 5 | 0.4976 |
| 6 | −0.0296 |
| 7 | −0.0758 |

**TABLE 7.3** Orthonormal fourth-order symlet filter coefficients for $h_\varphi(n)$. (Daubechies [1992].)

1-D symlets (i.e., wavelet and scaling functions). Figures 7.26(a) through (d) show the corresponding decomposition and reconstruction filters. The coefficients of lowpass reconstruction filter $g_0(n) = h_\varphi(n)$ for $0 \leq n \leq 7$ are given in Table 7.3. The coefficients of the remaining orthonormal filters are obtained using Eq. (7.1-14). Figure 7.26(g), a low-resolution graphic depiction of wavelet $\psi^V(x, y)$, is provided as an illustration of how a one-dimensional scaling and wavelet function can combine to form a separable, two-dimensional wavelet.

We conclude this section with two examples that demonstrate the usefulness of wavelets in image processing. As in the Fourier domain, the basic approach is to

**Step 1.** Compute a 2-D wavelet transform of an image.
**Step 2.** Alter the transform.
**Step 3.** Compute the inverse transform.

Because the DWT's scaling and wavelet vectors are used as lowpass and highpass filters, most Fourier-based filtering techniques have an equivalent "wavelet domain" counterpart.

**EXAMPLE 7.13:** Wavelet-based edge detection.

■ Figure 7.27 provides a simple illustration of the preceding three steps. In Fig. 7.27(a), the lowest scale approximation component of the discrete wavelet transform shown in Fig. 7.25(c) has been eliminated by setting its values to zero. As Fig. 7.27(b) shows, the net effect of computing the inverse wavelet transform using these modified coefficients is edge enhancement, reminiscent of the Fourier-based image sharpening results discussed in Section 4.9. Note how well the transitions between signal and background are delineated, despite the fact that they are relatively soft, sinusoidal transitions. By zeroing the horizontal details as well—see Figs. 7.27(c) and (d)—we can isolate the vertical edges. ■

a b
c d

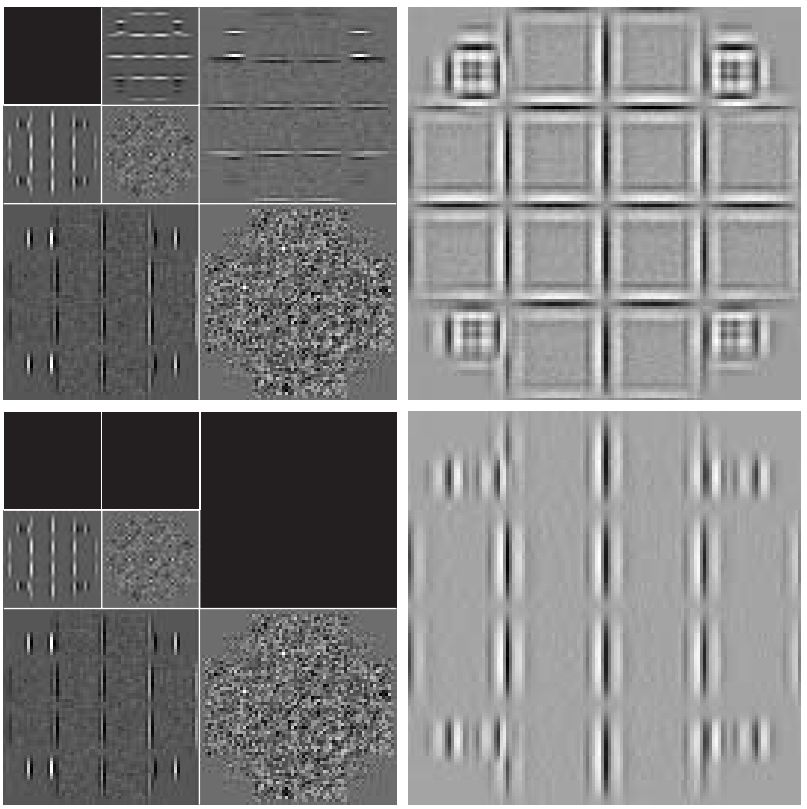


**FIGURE 7.27** Modifying a DWT for edge detection: (a) and (c) two-scale decompositions with selected coefficients deleted; (b) and (d) the corresponding reconstructions.

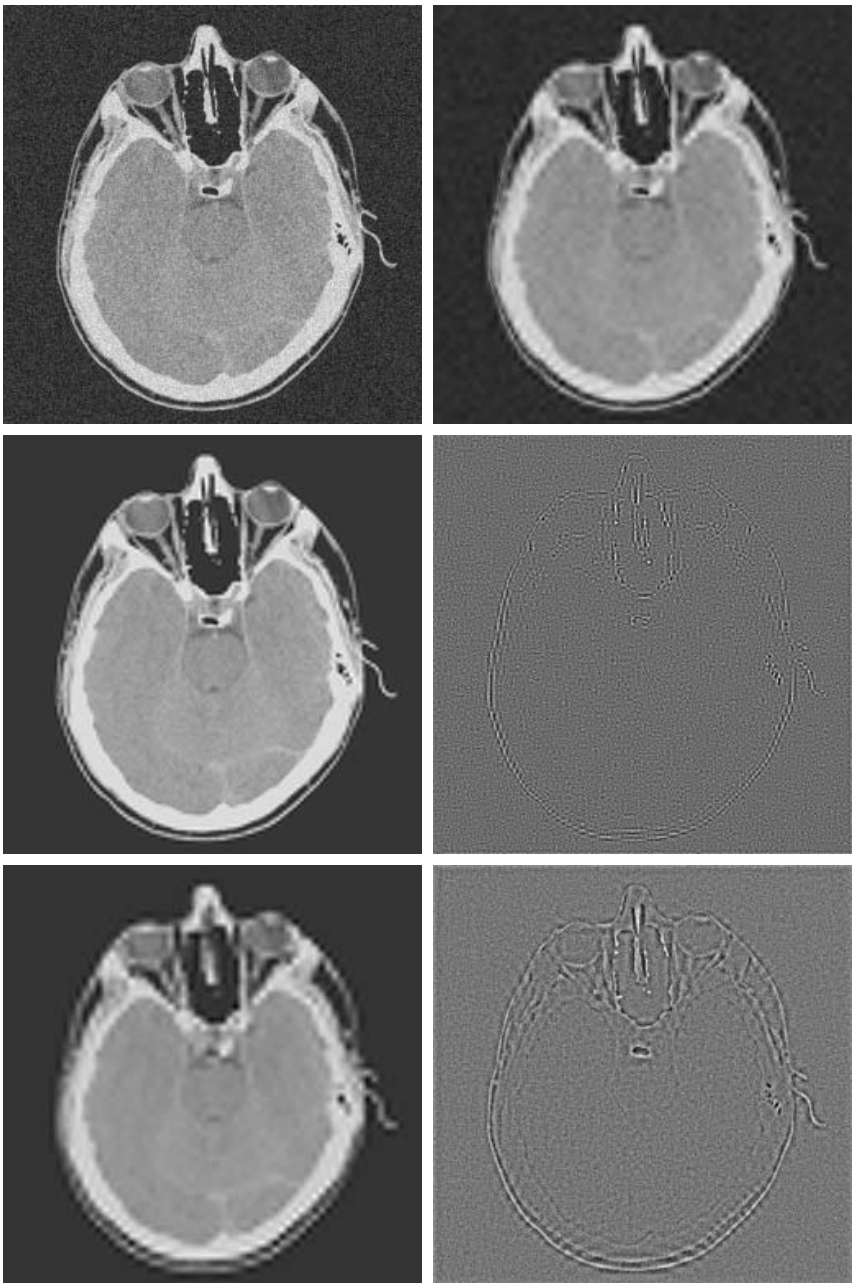
**EXAMPLE 7.14:** Wavelet-based noise removal.

■ As a second example, consider the CT image of a human head shown in Fig. 7.28(a). As can be seen in the background, the image has been uniformly corrupted with additive white noise. A general wavelet-based procedure for *denoising* the image (i.e., suppressing the noise part) is as follows:

**Step 1.** Choose a wavelet (e.g. Haar, symlet, ...) and number of levels (scales), $P$, for the decomposition. Then compute the FWT of the noisy image.

**Step 2.** Threshold the detail coefficients. That is, select and apply a threshold to the detail coefficients from scales $J - 1$ to $J - P$. This can be accomplished by *hard thresholding*, which means setting to zero the elements whose absolute values are lower than the threshold, or by *soft thresholding*, which involves first setting to zero the elements whose absolute values are lower than the threshold and then scaling the nonzero coefficients toward zero. Soft thresholding eliminates the discontinuity (at the threshold) that is inherent in hard thresholding. (See Chapter 10 for a discussion of thresholding.)

**Step 3.** Compute the inverse wavelet transform (i.e., perform a wavelet reconstruction) using the original approximation coefficients at level $J - P$ and the modified detail coefficients for levels $J - 1$ to $J - P$.



a b
c d
e f

**FIGURE 7.28**
Modifying a DWT for noise removal: (a) a noisy CT of a human head; (b), (c) and (e) various reconstructions after thresholding the detail coefficients; (d) and (f) the information removed during the reconstruction of (c) and (e). (Original image courtesy Vanderbilt University Medical Center.)

Figure 7.28(b) shows the result of performing these operations with fourth-order symlets, two scales (i.e., $P = 2$), and a global threshold that was determined interactively. Note the reduction in noise and blurring of image edges. This loss of edge detail is reduced significantly in Fig. 7.28(c), which

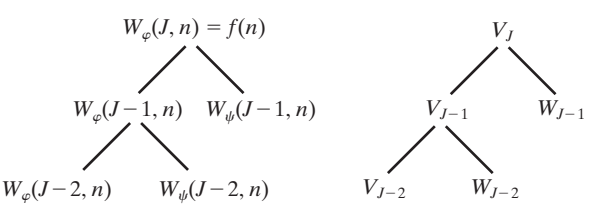
was generated by simply zeroing the highest-resolution detail coefficients (not thresholding the lower-resolution details) and reconstructing the image. Here, almost all of the background noise has been eliminated and the edges are only slightly disturbed. The difference image in Fig. 7.28(d) shows the information that is lost in the process. This result was generated by computing the inverse FWT of the two-scale transform with all but the highest-resolution detail coefficients zeroed. As can be seen, the resulting image contains most of the noise in the original image and some of the edge information. Figures 7.28(e) and (f) are included to show the negative effect of deleting all the detail coefficients. That is, Fig. 7.28(e) is a reconstruction of the DWT in which the details at both levels of the two-scale transform have been zeroed; Fig. 7.28(f) shows the information that is lost. Note the significant increase in edge information in Fig. 7.28(f) and the corresponding decrease in edge detail in Fig. 7.28(e). ■

Because only the highest resolution detail coefficients were kept when generating Fig. 7.28(d), the inverse transform is their contribution to the image. In the same way, Fig. 7.28(f) is the contribution of all the detail coefficients.

## 7.6 Wavelet Packets

The fast wavelet transform decomposes a function into a sum of scaling and wavelet functions whose bandwidths are logarithmically related. That is, the low frequency content (of the function) is represented using (scaling and wavelet) functions with narrow bandwidths, while the high-frequency content is represented using functions with wider bandwidths. If you look along the frequency axis of the time-frequency plane in Fig. 7.23(c), this is immediately apparent. Each horizontal strip of constant height tiles, which contains the basis functions for a single FWT scale, increases logarithmically in height as you move up the frequency axis. If we want greater control over the partitioning of the time-frequency plane (e.g., smaller bands at the higher frequencies), the FWT must be generalized to yield a more flexible decomposition—called a *wavelet packet* (Coifman and Wickerhauser [1992]). The cost of this generalization is an increase in computational complexity from $O(M)$ for the FWT to $O(M \log_2 M)$ for a wavelet packet.

Consider again the two-scale filter bank of Fig. 7.18(a)—but imagine the decomposition as a *binary tree*. Figure 7.29(a) details the structure of the tree, and links the appropriate FWT scaling and wavelet coefficients [from Fig. 7.18(a)] to its *nodes*. The *root node* is assigned the highest-scale approximation coefficients,



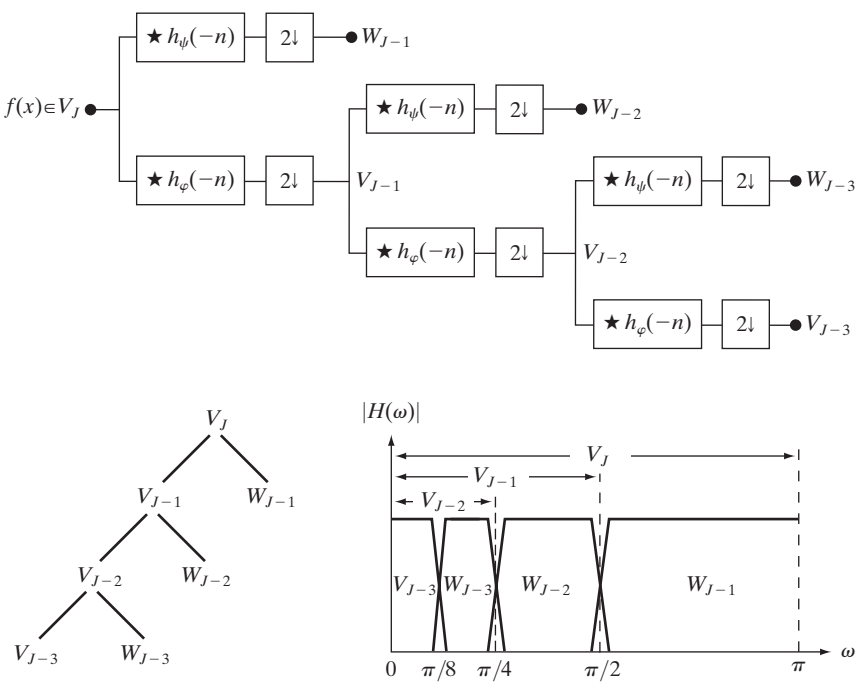

a b

**FIGURE 7.29** An (a) coefficient tree and (b) analysis tree for the two-scale FWT analysis bank of Fig. 7.18.

which are samples of the function itself, while the *leaves* inherit the transform's approximation and detail coefficient outputs. The lone intermediate node, $W_\varphi(J-1, n)$, is a filter bank approximation that is ultimately filtered to become two leaf nodes. Note that the coefficients of each node are the weights of a linear expansion that produces a band-limited "piece" of root node $f(n)$. Because any such piece is an element of a known scaling or wavelet subspace (see Sections 7.2.2 and 7.2.3), we can replace the generating coefficients in Fig. 7.29(a) by the corresponding subspace. The result is the *subspace analysis tree* of Fig. 7.29(b). Although the variable $W$ is used to denote both coefficients and subspaces, the two quantities are distinguishable by the format of their subscripts.

These concepts are further illustrated in Fig. 7.30, where a three-scale FWT analysis bank, analysis tree, and corresponding frequency spectrum are depicted. Unlike Fig. 7.18(a), the block diagram of Fig. 7.30(a) is labeled to resemble the analysis tree in Fig. 7.30(b)—as well as the spectrum in Fig. 7.30(c). Thus, while the output of the upper-left filter and subsampler is, to be accurate, $W_\psi(J-1, n)$, it has been labeled $W_{J-1}$—the subspace of the function that is generated by the $W_\psi(J-1, n)$ transform coefficients. This subspace corresponds to the upper-right leaf of the associated analysis tree, as well as the rightmost (widest bandwidth) segment of the corresponding frequency spectrum.

Analysis trees provide a compact and informative way of representing multiscale wavelet transforms. They are simple to draw, take less space than their corresponding filter and subsampler-based block diagrams, and make it relatively




a
b c

**FIGURE 7.30** A three-scale FWT filter bank: (a) block diagram; (b) decomposition space tree; and (c) spectrum splitting characteristics.

easy to detect valid decompositions. The three-scale analysis tree of Fig. 7.30(b), for example, makes possible the following three expansion options:

$$V_J = V_{J-1} \oplus W_{J-1} \tag{7.6-1}$$

$$V_J = V_{J-2} \oplus W_{J-2} \oplus W_{J-1} \tag{7.6-2}$$

$$V_J = V_{J-3} \oplus W_{J-3} \oplus W_{J-2} \oplus W_{J-1} \tag{7.6-3}$$

They correspond to the one-, two-, and three-scale FWT decompositions of Section 7.4 and may be obtained from Eq. (7.2-27) of Section 7.2.3 by letting $j_0 = J - P$ for $P = \{1, 2, 3\}$. In general, a $P$-scale FWT analysis tree supports $P$ unique decompositions.

Analysis trees also are an efficient mechanism for representing *wavelet packets*, which are nothing more than *conventional wavelet transforms in which the details are filtered iteratively*. Thus, the three-scale FWT analysis tree of Fig. 7.30(b) becomes the three-scale *wavelet packet* tree of Fig. 7.31. Note the additional subscripting that is introduced. The first subscript of a double-subscripted node identifies the scale of the FWT *parent* node from which it descended. The second—a variable length string of $A$s and $D$s—encodes the path from the parent to the node. An $A$ designates approximation filtering, while a $D$ indicates detail filtering. Subspace $W_{J-1,DA}$, for example, is obtained by "filtering" the scale $J - 1$ FWT coefficients (i.e., parent $W_{J-1}$ in Fig. 7.31) through an additional detail filter (yielding $W_{J-1,D}$), followed by an approximation filter (giving $W_{J-1,DA}$). Figures 7.32(a) and (b) are the filter bank and spectrum splitting characteristics of the analysis tree in Fig. 7.31. Note that the "naturally ordered" outputs of the filter bank in Fig. 7.32(a) have been reordered based on frequency content in Fig. 7.32(b) (see Problem 7.25 for more on "frequency ordered" wavelets).

The three-scale packet tree in Fig. 7.31 almost triples the number of decompositions (and associated time-frequency tilings) that are available from the three-scale FWT tree. Recall that in a normal FWT, we split, filter, and downsample the lowpass bands alone. This creates a fixed logarithmic (base 2) relationship between the bandwidths of the scaling and wavelet spaces used in the representation of a function [see Figure 7.30(c)]. Thus, while the three-scale FWT analysis tree of Fig. 7.30(a) offers three possible decompositions—defined by Eqs. (7.6-1) to (7.6-3)—the wavelet packet tree of Fig. 7.31 supports 26 different decompositions. For instance, $V_J$ [and therefore function $f(n)$] can be expanded as

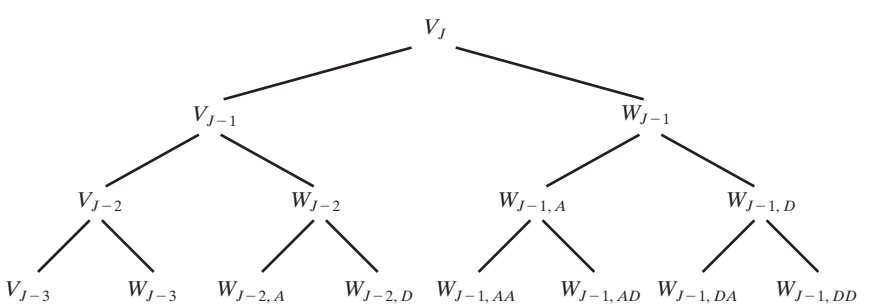
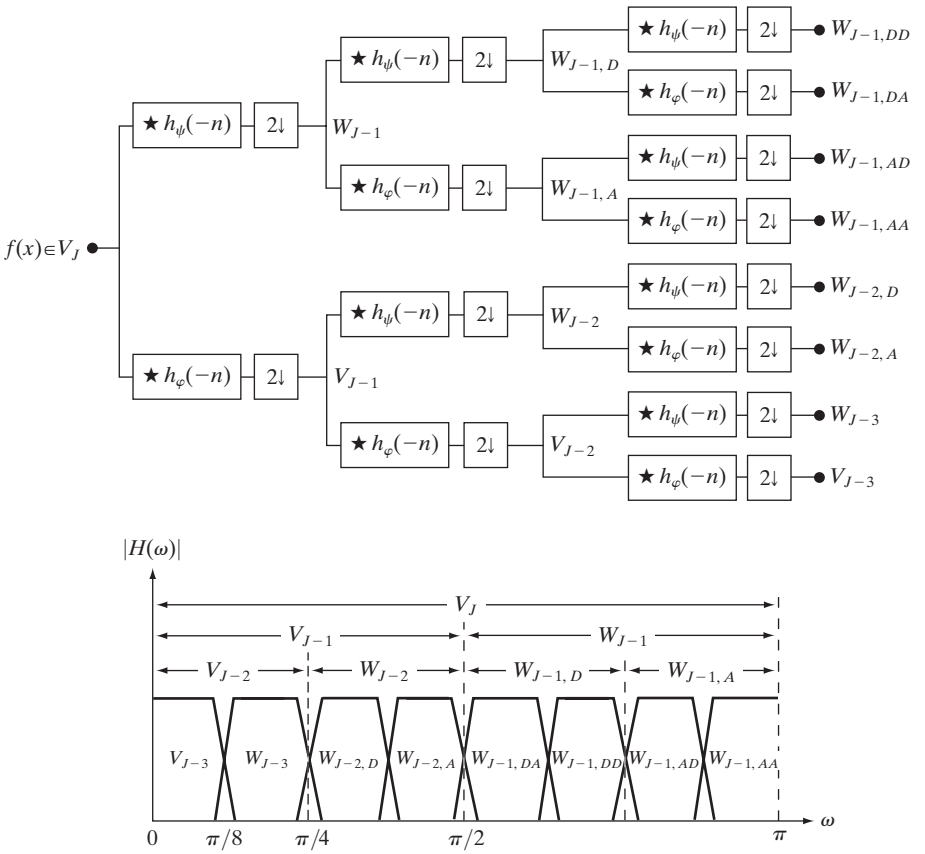



**FIGURE 7.31**
A three-scale wavelet packet analysis tree.




a
b

**FIGURE 7.32** The (a) filter bank and (b) spectrum splitting characteristics of a three-scale full wavelet packet analysis tree.

$$V_J = V_{J-3} \oplus W_{J-3} \oplus W_{J-2,A} \oplus W_{J-2,D} \oplus W_{J-1,AA} \oplus W_{J-1,AD} \oplus W_{J-1,DA} \oplus W_{J-1,DD} \quad (7.6\text{-}4)$$

Recall that $\oplus$ denotes the union of spaces (like the union of sets). The 26 decompositions associated with Fig. 7.31 are determined by various combinations of nodes (spaces) that can be combined to represent the root node (space) at the top of the tree. Eqs. (7.6-4) and (7.6-5) define two of them.

whose spectrum is shown in Fig. 7.32(b), or

$$V_J = V_{J-1} \oplus W_{J-1,A} \oplus W_{J-1,DA} \oplus W_{J-1,DD} \quad (7.6\text{-}5)$$

whose spectrum is depicted in Fig. 7.33. Note the difference between this last spectrum and the full packet spectrum of Fig. 7.32(b), or the three-scale FWT

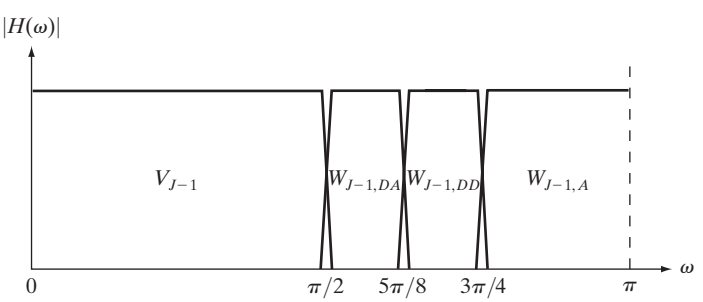



**FIGURE 7.33** The spectrum of the decomposition in Eq. (7.6-5).

spectrum of Fig. 7.30(c). In general, $P$-scale, one-dimensional wavelet packet transforms (and associated $P + 1$-level analysis trees) support

$$D(P + 1) = \left[D(P)\right]^2 + 1 \tag{7.6-6}$$

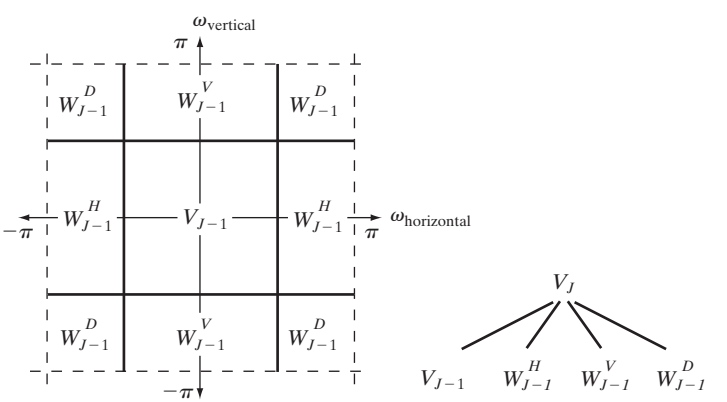
unique decompositions, where $D(1) = 1$. With such a large number of valid expansions, packet-based transforms provide improved control over partitioning the spectrum of the decomposed function. The cost of this control is an increase in computational complexity [compare the filter bank in Fig. 7.30(a) to that of Fig. 7.32(a)].

Now consider the two-dimensional, four-band filter bank of Fig. 7.24(a). As was noted in Section 7.5, it splits approximation $W_\varphi(j + 1, m, n)$ into outputs, $W_\varphi(j, m, n)$, $W_\psi^H(j, m, n)$, $W_\psi^V(j, m, n)$, and $W_\psi^D(j, m, n)$. As in the one-dimensional case, it can be "iterated" to generate $P$ scale transforms for scales $j = J - 1, J - 2, \ldots, J - P$, with $W_\varphi(J, m, n) = f(m, n)$. The spectrum resulting from the first iteration [i.e., using $j + 1 = J$ in Fig. 7.24(a)] is shown in Fig. 7.34(a). Note that it divides the frequency plane into four equal areas. The low-frequency quarter-band in the center of the plane coincides with transform coefficients $W_\varphi(J - 1, m, n)$ and scaling space $V_{J-1}$. (This nomenclature is consistent with the one-dimensional case.) To accommodate the two-dimensional nature of the input, however, we now have three (rather than one) wavelet subspaces. They are denoted $W_{J-1}^H$, $W_{J-1}^V$, and $W_{J-1}^D$ and correspond to coefficients $W_\psi^H(J - 1, m, n)$, $W_\psi^V(J - 1, m, n)$, and $W_\psi^D(J - 1, m, n)$, respectively. Figure 7.34(b) shows the resulting four-band, single-scale *quaternary FWT analysis tree*. Note the superscripts that link the wavelet subspace designations to their transform coefficient counterparts.

Figure 7.35 shows a portion of a three-scale, two-dimensional wavelet packet analysis tree. Like its one-dimensional counterpart in Fig. 7.31, the first subscript of every node that is a descendant of a conventional FWT detail node is the scale of that *parent* detail node. The second subscript—a variable length string of *A*s, *H*s, *V*s, and *D*s—encodes the path from the parent to the node under consideration. The node labeled $W_{J-1,VD}^H$, for example, is obtained by "row/column filtering" the

a b

**FIGURE 7.34** The first decomposition of a two-dimensional FWT: (a) the spectrum and (b) the subspace analysis tree.



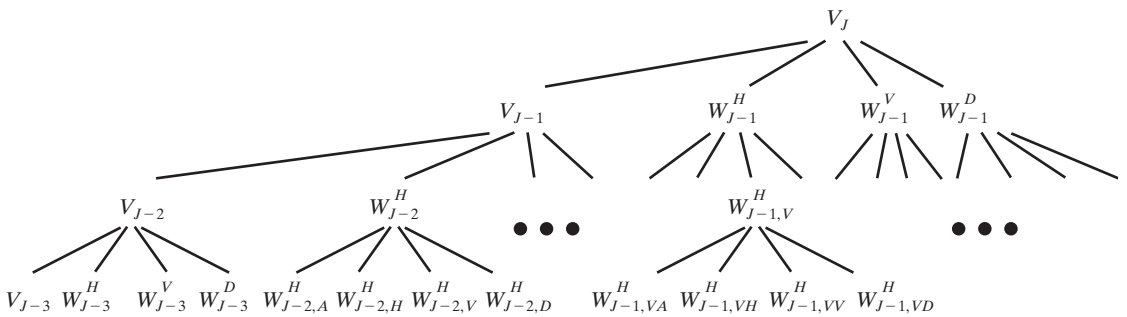



**FIGURE 7.35** A three-scale, full wavelet packet decomposition tree. Only a portion of the tree is provided.

scale $J - 1$ FWT horizontal detail coefficients (i.e., parent $W^H_{J-1}$ in Fig. 7.35) through an additional detail/approximation filter (yielding $W^H_{J-1,V}$), followed by a detail/detail filter (giving $W^H_{J-1,VD}$). A $P$-scale, two-dimensional wavelet packet tree supports

$$D(P + 1) = \left[D(P)\right]^4 + 1 \tag{7.6-7}$$

unique expansions, where $D(1) = 1$. Thus, the three-scale tree of Fig. 7.35 offers 83,522 possible decompositions. The problem of selecting among them is the subject of the next example.

**EXAMPLE 7.15:** Two-dimensional wavelet packet decompositions.

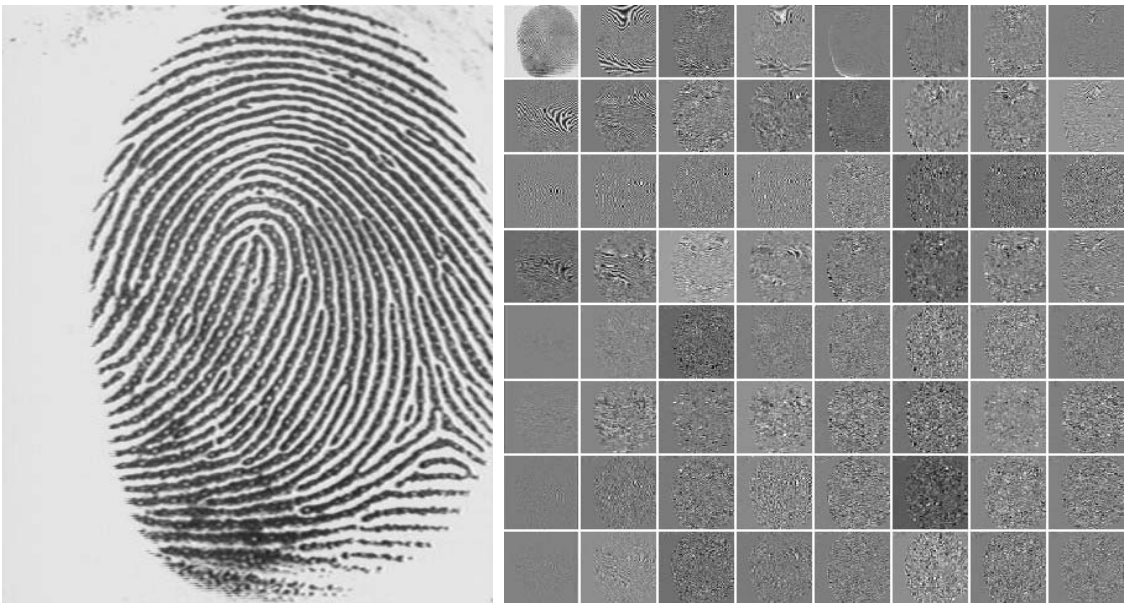
■ As noted in the above discussion, a single wavelet packet tree presents numerous decomposition options. In fact, the number of possible decompositions is often so large that it is impractical, if not impossible, to enumerate or examine them individually. An efficient algorithm for finding optimal decompositions with respect to application specific criteria is highly desirable. As will be seen, classical entropy- and energy-based cost functions are applicable in many situations and are well suited for use in binary and quaternary tree searching algorithms.

Consider the problem of reducing the amount of data needed to represent the $400 \times 480$ fingerprint image in Fig. 7.36(a). Image compression is discussed in detail in Chapter 8. In this example, we want to select the "best" three-scale wavelet packet decomposition as a starting point for the compression process. Using three-scale wavelet packet trees, there are 83,522 [see Eq. (7.6-7)] potential decompositions. Figure 7.36(b) shows one of them—a full wavelet packet, 64-leaf decomposition like the analysis tree of Fig. 7.35. Note that the leaves of the tree correspond to the subbands of the $8 \times 8$ array of decomposed subimages in Fig. 7.36(b). The probability that this particular 64-leaf decomposition is in some way optimal for the purpose of compression, however, is relatively low. In the absence of a suitable optimality criterion, we can neither confirm nor deny it.

The 64 leaf nodes in Fig. 7.35 correspond to the $8 \times 8$ array of 64 subimages in Fig. 7.36(b). Despite appearances, they are not square. The distortion (particularly noticeable in the approximation subimage) is due to the program used to produce the result.

One reasonable criterion for selecting a decomposition for the compression of the image of Fig. 7.36(a) is the additive cost function

$$E(f) = \sum_{m,n} \left| f(m, n) \right| \tag{7.6-8}$$



a b

**FIGURE 7.36** (a) A scanned fingerprint and (b) its three-scale, full wavelet packet decomposition. (Original image courtesy of the National Institute of Standards and Technology.)

Other possible energy measures include the sum of the squares of $f(x, y)$, the sum of the log of the squares, etc. Problem 7.27 defines one possible entropy-based cost function.

This function provides one possible measure of the energy content of two-dimensional function $f$. Under this measure, the energy of function $f(m, n) = 0$ for all $m$ and $n$ is 0. High values of $E$, on the other hand, are indicative of functions with many nonzero values. Since most transform-based compression schemes work by truncating or thresholding the small coefficients to zero, a cost function that maximizes the number of near-zero values is a reasonable criterion for selecting a "good" decomposition from a compression point of view.

The cost function just described is both computationally simple and easily adapted to tree optimization routines. The optimization algorithm must use the function to minimize the "cost" of the leaf nodes in the decomposition tree. Minimal energy leaf nodes should be favored because they have more near-zero values, which leads to greater compression. Because the cost function of Eq. (7.6-8) is a local measure that uses only the information available at the node under consideration, an efficient algorithm for finding minimal energy solutions is easily constructed as follows:

For each node of the analysis tree, beginning with the root and proceeding level by level to the leaves:

**Step 1.** Compute both the energy of the node, denoted $E_P$ (for parent energy), and the energy of its four offspring—denoted $E_A$, $E_H$, $E_V$, and $E_D$. For two-dimensional wavelet packet decompositions, the parent is a two-dimensional array of approximation or detail coefficients; the

offspring are the filtered approximation, horizontal, vertical, and diagonal details.

**Step 2.** If the combined energy of the offspring is less than the energy of the parent—that is, $E_A + E_H + E_V + E_D < E_P$—include the offspring in the analysis tree. If the combined energy of the offspring is greater than or equal to that of the parent, prune the offspring, keeping only the parent. It is a leaf of the optimized analysis tree.

The preceding algorithm can be used to (1) prune wavelet packet trees or (2) design procedures for computing optimal trees from scratch. In the latter case, nonessential siblings—descendants of nodes that would be eliminated in step 2 of the algorithm—would not be computed. Figure 7.37 shows the optimized decomposition that results from applying the algorithm to the image of Fig. 7.36(a) with the cost function of Eq. (7.6-8). The corresponding analysis tree is given in Fig. 7.38. Note that many of the original full packet decomposition's 64 subbands in Fig. 7.36(b) (and corresponding 64 leaves of the analysis tree in Fig. 7.35) have been eliminated. In addition, the subimages that are not split (further decomposed) in Fig. 7.37 are relatively smooth and composed of pixels that are middle gray in value. Because all but the approximation subimage of this figure have been scaled so that gray level 128 indicates a zero-valued coefficient, these subimages contain little energy. There would be no overall decrease in energy realized by splitting them. ■

The preceding example is based on a real-world problem that was solved through the use of wavelets. The Federal Bureau of Investigation (FBI) currently maintains a large database of fingerprints and has established a wavelet-based national standard for the digitization and compression of fingerprint

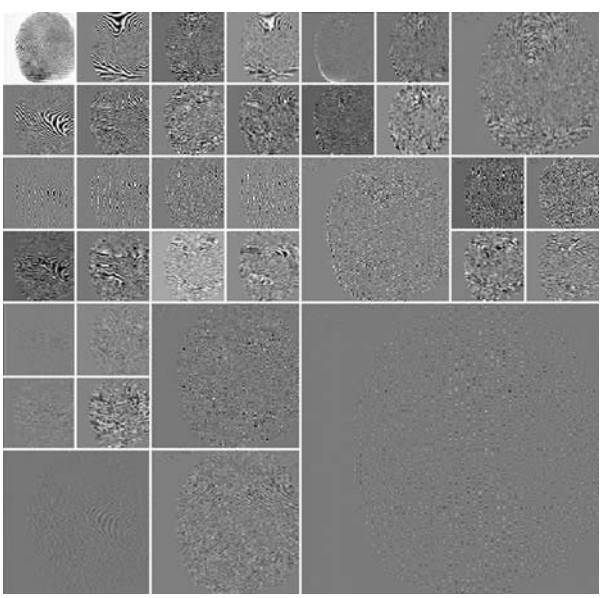


**FIGURE 7.37** An optimal wavelet packet decomposition for the fingerprint of Fig. 7.36(a).

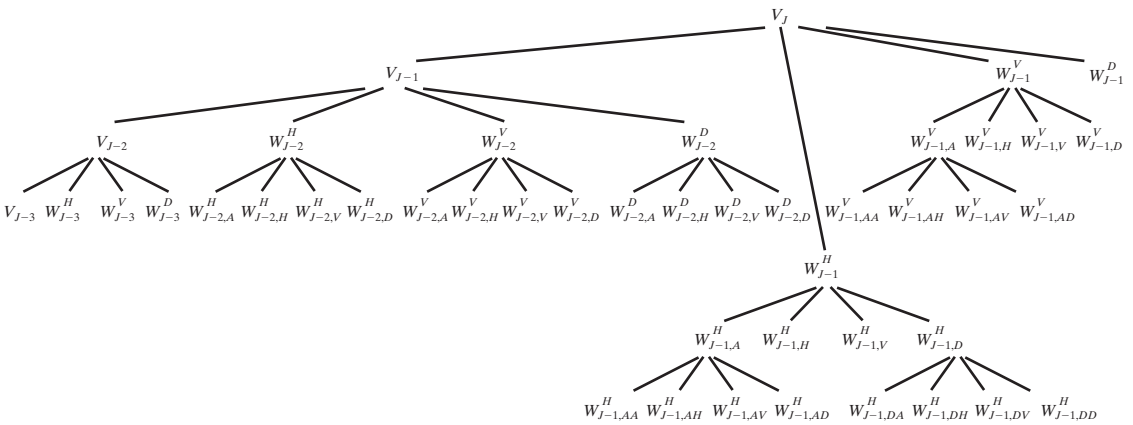


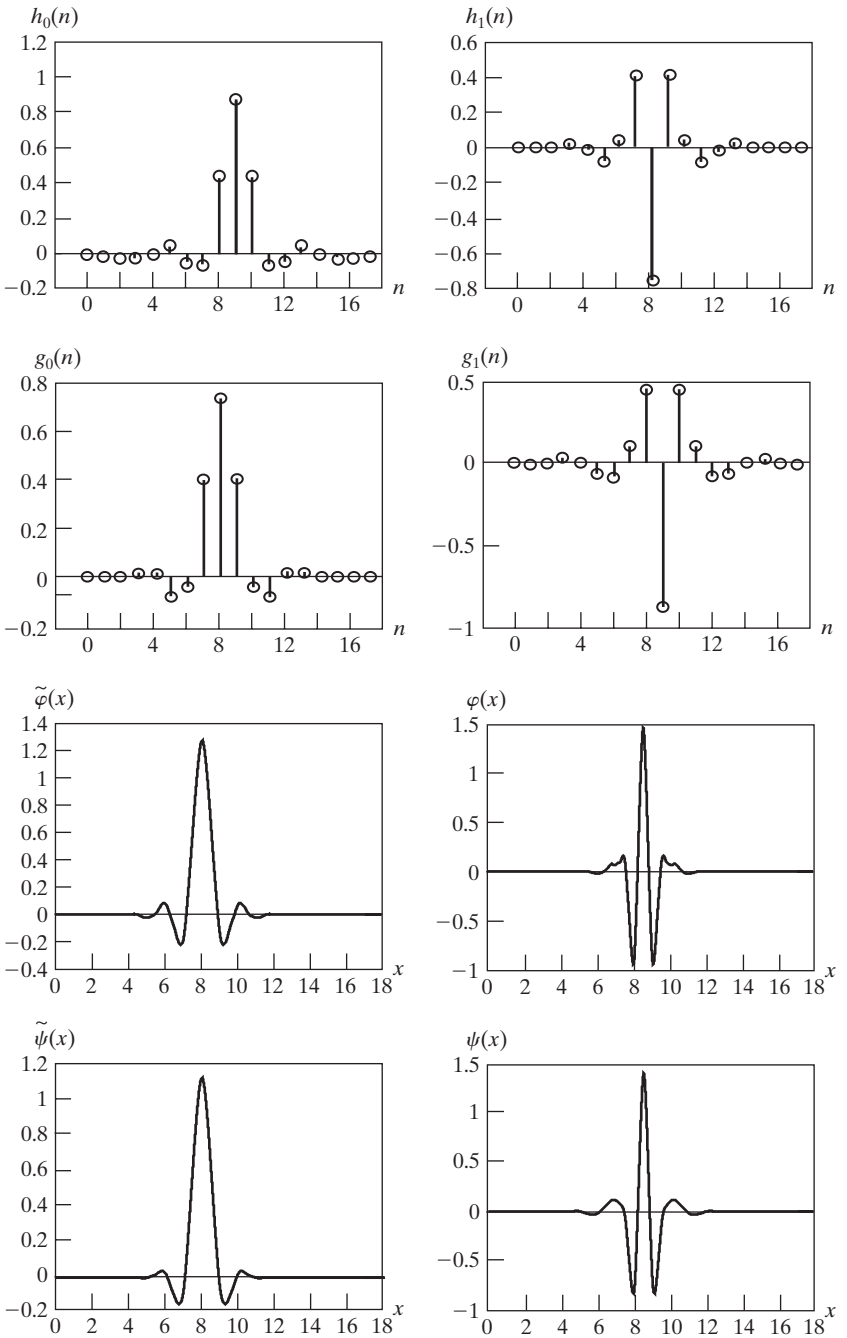

**FIGURE 7.38** The optimal wavelet packet analysis tree for the decomposition in Fig. 7.37.

images (FBI [1993]). Using biorthogonal wavelets, the standard achieves a typical compression ratio of 15:1. The advantages of wavelet-based compression over the more traditional JPEG approach are examined in the next chapter.

The decomposition filters used in Example 7.15, as well as by the FBI, are part of a well-known family of wavelets called Cohen-Daubechies-Feauveau biorthogonal wavelets (Cohen, Daubechies, and Feauveau [1992]). Because the scaling and wavelet functions of the family are symmetrical and have similar lengths, they are among the most widely used biorthogonal wavelets. Figures 7.39(e) through (h) show the dual scaling and wavelet functions. Figures 7.39(a) through (d) are the corresponding decomposition and reconstruction filters. The coefficients of the lowpass and highpass decomposition filters, $h_0(n)$ and $h_1(n)$ for $0 \le n \le 17$ are shown in Table 7.4. The corresponding coefficients of the biorthogonal synthesis filters can be computed using $g_0(n) = (-1)^{n+1}h_1(n)$ and $g_1(n) = (-1)^n h_0(n)$ of Eq. (7.1-11). That is, they are cross-modulated versions of the decomposition filters. Note that zero padding is employed to make the filters the same length and that Table 7.4 and Fig. 7.39 define them with respect to the subband coding and decoding system of Fig. 7.6(a); with respect to the FWT, $h_\varphi(-n) = h_0(n)$ and $h_\psi(-n) = h_1(n)$.

**TABLE 7.4** Biorthogonal Cohen-Daubechies-Feauveau filter coefficients (Cohen, Daubechies, and Feauveau [1992]).

| $n$ | $h_0(n)$ | $h_1(n)$ | $n$ | $h_0(n)$ | $h_1(n)$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 9 | 0.8259 | 0.4178 |
| 1 | 0.0019 | 0 | 10 | 0.4208 | 0.0404 |
| 2 | −0.0019 | 0 | 11 | −0.0941 | −0.0787 |
| 3 | −0.017 | 0.0144 | 12 | −0.0773 | −0.0145 |
| 4 | 0.0119 | −0.0145 | 13 | 0.0497 | 0.0144 |
| 5 | 0.0497 | −0.0787 | 14 | 0.0119 | 0 |
| 6 | −0.0773 | 0.0404 | 15 | −0.017 | 0 |
| 7 | −0.0941 | 0.4178 | 16 | −0.0019 | 0 |
| 8 | 0.4208 | −0.7589 | 17 | 0.0010 | 0 |




a b
c d
e f
g h

**FIGURE 7.39** A member of the Cohen-Daubechies-Feauveau biorthogonal wavelet family: (a) and (b) decomposition filter coefficients; (c) and (d) reconstruction filter coefficients; (e)–(h) dual wavelet and scaling functions. See Table 7.3 for the values of $h_0(n)$ and $h_1(n)$ for $0 \leq n \leq 17$.

# *Summary*

The material of this chapter establishes a solid mathematical foundation for understanding and accessing the role of wavelets and multiresolution analysis in image processing. Wavelets and wavelet transforms are relatively new imaging tools that are being rapidly applied to a wide variety of image processing problems. Because of their similarity to the Fourier transform, many of the techniques in Chapter 4 have wavelet domain counterparts. A partial listing of the imaging applications that have been approached from a wavelet point of view includes image matching, registration, segmentation, denoising, restoration, enhancement, compression, morphological filtering, and computed tomography. Since it is impractical to cover all of these applications in a single chapter, the topics included were chosen for their value in introducing or clarifying fundamental concepts and preparing the reader for further study in the field. In Chapter 8, we will apply wavelets to the compression of images.

## References and Further Reading

There are many good texts on wavelets and their application. Several complement our treatment and were relied upon during the development of the core sections of the chapter. The material in Section 7.1.2 on subband coding and digital filtering follows the book by Vetterli and Kovacevic [1995], while Sections 7.2 and 7.4 on multiresolution expansions and the fast wavelet transform follow the treatment of these subjects in Burrus, Gopinath, and Guo [1998]. The remainder of the material in the chapter is based on the references cited in the text. All of the examples in the chapter were done using MATLAB (see Gonzalez et al. [2004]).

The history of wavelet analysis is recorded in a book by Hubbard [1998]. The early predecessors of wavelets were developed simultaneously in different fields and unified in a paper by Mallat [1987]. It brought a mathematical framework to the field. Much of the history of wavelets can be traced through the works of Meyer [1987] [1990] [1992a, 1992b] [1993], Mallat [1987] [1989a–c] [1998], and Daubechies [1988] [1990] [1992] [1993] [1996]. The current interest in wavelets was stimulated by many of their publications. The book by Daubechies [1992] is a classic source for the mathematical details of wavelet theory.

The application of wavelets to image processing is addressed in general image processing texts, like Castleman [1996], and many application specific books, some of which are conference proceedings. In this latter category, for example, are Rosenfeld [1984], Prasad and Iyengar [1997], and Topiwala [1998]. Recent articles that can serve as starting points for further research into specific imaging applications include Gao et al. [2007] on corner detection; Olkkonen and Olkkonen [2007] on lattice implementations; Selesnick et al. [2005] and Kokare et al. [2005] on complex wavelets; Thévenaz and Unser [2000] for image registration; Chang and Kuo [1993] and Unser [1995] on texture-based classification; Heijmans and Goutsias [2000] on morphological wavelets; Banham et al. [1994], Wang, Zhang, and Pan [1995], and Banham and Kastaggelos [1996] on image restoration; Xu et al. [1994] and Chang, Yu, and Vetterli [2000] on image enhancement; Delaney and Bresler [1995] and Westenberg and Roerdink [2000] on computed tomography; and Lee, Sun, and Chen [1995], Liang and Kuo [1999], Wang, Lee, and Toraichi [1999], and You and Bhattacharya [2000] on image description and matching. One of the most important applications of wavelets is image compression—see, for example, Brechet et al. [2007], Demin Wang et al. [2006], Antonini et al. [1992], Wei et al. [1998], and the book by Topiwala [1998]. Finally, there have been a number of special issues devoted to wavelets, including a special issue on wavelet transforms and multiresolution signal anaysis in the *IEEE Transactions on Information Theory* [1992], a special issue on wavelets and signal processing in the *IEEE Transactions on Signal*

*Processing* [1993], and a special section on multiresolution representation in the *IEEE Transactions on Pattern Analysis and Machine Intelligence* [1989].

Although the chapter focuses on the fundamentals of wavelets and their application to image processing, there is considerable interest in the construction of wavelets themselves. The interested reader is referred to the work of Battle [1987] [1988], Daubechies [1988] [1992], Cohen and Daubechies [1992], Meyer [1990], Mallat [1989b], Unser, Aldroubi, and Eden [1993], and Gröchenig and Madych [1992]. This is not an exhaustive list but should serve as a starting point for further reading. See also the general references on subband coding and filter banks, including Strang and Nguyen [1996] and Vetterli and Kovacevic [1995], and the references included in the chapter with respect to the wavelets we used as examples.

## Problems

**7.1** Design a system for decoding the prediction residual pyramid generated by the encoder of Fig. 7.2(b) and draw its block diagram. Assume there is no quantization error introduced by the encoder.

**★7.2** Construct a fully populated approximation pyramid and corresponding prediction residual pyramid for the image

$$f(x, y) = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

Use $2 \times 2$ block neighborhood averaging for the approximation filter in Fig. 7.2(b) and assume the interpolation filter implements pixel replication.

**★7.3** Given a $2^J \times 2^J$ image, does a $J + 1$-level pyramid reduce or expand the amount of data required to represent the image? What is the compression or expansion ratio?

**7.4** Is the two-band subband coding filter bank containing filters $h_0(n) = \left\{1/\sqrt{2}, 1/\sqrt{2}\right\}$, $h_1(n) = \left\{-1/\sqrt{2}, 1/\sqrt{2}\right\}$, $g_0(n) = \left\{1/\sqrt{2}, 1/\sqrt{2}\right\}$, and $g_1(n) = \left\{1/\sqrt{2}, -1/\sqrt{2}\right\}$ orthonormal, biorthogonal, or both?

**7.5** Given the sequence $f(n) = \{0, 0.5, 0.25, 1\}$ where $n = 0, 1, 2, 3$, compute:

**(a)** The sign-reversed sequence.

**(b)** The order-reversed sequence.

**(c)** The modulated sequence.

**(d)** The modulated and then order-reversed sequence.

**(e)** The order-reversed and then modulated sequence.

**(f)** Does the result from (d) or (e) correspond to Eq. (7.1-9)?

**7.6** Compute the coefficients of the Daubechies synthesis filters $g_0(n)$ and $g_1(n)$ for Example 7.2. Using Eq. (7.1-13) with $m = 0$ only, show that the filters are orthonormal.

**★7.7** Draw a two-dimensional four-band filter bank decoder to reconstruct input $f(m, n)$ in Fig. 7.7.

**7.8** Obtain the Haar transformation matrix for $N = 8$.

**7.9** **(a)** Compute the Haar transform of the $2 \times 2$ image

$$\mathbf{F} = \begin{bmatrix} 3 & -1 \\ 6 & 2 \end{bmatrix}$$

**(b)** The inverse Haar transform is $\mathbf{F} = \mathbf{H}^T\mathbf{T}\mathbf{H}$, where $\mathbf{T}$ is the Haar transform of $\mathbf{F}$ and $\mathbf{H}^T$ is the matrix inverse of $\mathbf{H}$. Show that $\mathbf{H}_2^{-1} = \mathbf{H}_2^T$ and use it to compute the inverse Haar transform of the result in (a).

**7.10** Compute the expansion coefficients of 2-tuple $[3, 2]^T$ for the following bases and write the corresponding expansions:

★**(a)** Basis $\varphi_0 = \left[1/\sqrt{2}, 1/\sqrt{2},\right]^T$ and $\varphi_1 = \left[1/\sqrt{2}, -1/\sqrt{2},\right]^T$ on $\mathbf{R}^2$, the set of real 2-tuples.

**(b)** Basis $\varphi_0 = [1, 0]^T$ and $\varphi_1 = [1, 1]^T$, and its dual, $\widetilde{\varphi}_0 = [1, -1]^T$ and $\widetilde{\varphi}_1 = [0, 1]^T$, on $\mathbf{R}^2$.

**(c)** Basis $\varphi_0 = [1, 0]^T$, $\varphi_1 = \left[-1/2, \sqrt{3}/2\right]^T$, and $\varphi_2 = \left[-1/2, -\sqrt{3}/2\right]^T$, and their duals, $\widetilde{\varphi}_i = 2\varphi_i/3$ for $i = \{0, 1, 2,\}$, on $\mathbf{R}^2$.

(*Hint*: Vector inner products must be used in place of the integral inner products of Section 7.2.1.)

**7.11** Show that scaling function

$$\varphi(x) = \begin{cases} 1 & 0.25 \le x < 0.75 \\ 0 & \text{elsewhere} \end{cases}$$

does not satisfy the second requirement of a multiresolution analysis.

**7.12** Write an expression for scaling space $V_3$ as a function of scaling function $\varphi(x)$. Use the Haar scaling function definition of Eq. (7.2-14) to draw the Haar $V_3$ scaling functions at translations $k = \{0, 1, 2\}$.

★**7.13** Draw wavelet $\psi_{3,3}(x)$ for the Haar wavelet function. Write an expression for $\psi_{3,3}(x)$ in terms of Haar scaling functions.

**7.14** Suppose function $f(x)$ is a member of Haar scaling space $V_3$—that is, $f(x) \in V_3$. Use Eq. (7.2-22) to express $V_3$ as a function of scaling space $V_0$ and any required wavelet spaces. If $f(x)$ is 0 outside the interval $[0, 1)$, sketch the scaling and wavelet functions required for a linear expansion of $f(x)$ based on your expression.

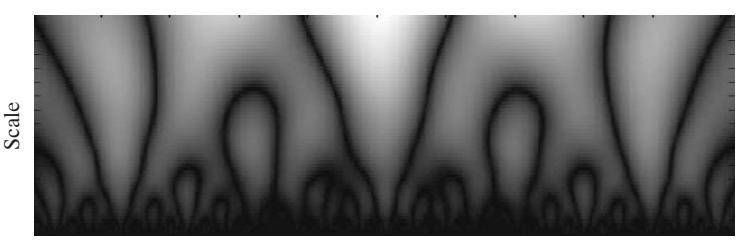
**7.15** Compute the first four terms of the wavelet series expansion of the function used in Example 7.7 with starting scale $j_0 = 1$. Write the resulting expansion in terms of the scaling and wavelet functions involved. How does your result compare to the example, where the starting scale was $j_0 = 0$?

**7.16** The DWT in Eqs. (7.3-5) and (7.3-6) is a function of starting scale $j_0$.

**(a)** Recompute the one-dimensional DWT of function $f(n) = \{1, 4, -3, 0\}$ for $0 \le n \le 3$ in Example 7.8 with $j_0 = 1$ (rather than 0).

**(b)** Use the result from (a) to compute $f(1)$ from the transform values.

★**7.17** What does the following continuous wavelet transform reveal about the one-dimensional function upon which it was based?



**7.18** **(a)** The continuous wavelet transform of Problem 7.17 is computer generated. The function upon which it is based was first sampled at discrete intervals. What is continuous about the transform—or what distinguishes it from the discrete wavelet transform of the function?

★ **(b)** Under what circumstances is the DWT a better choice than the CWT? Are there times when the CWT is better than the DWT?

★**7.19** Draw the FWT filter bank required to compute the transform in Problem 7.16. Label all inputs and outputs with the appropriate sequences.

**7.20** The computational complexity of an $M$-point fast wavelet transform is $O(M)$. That is, the number of operations is proportional $M$. What determines the constant of proportionality?

**7.21** ★ **(a)** If the input to the three-scale FWT filter bank of Fig. 7.30(a) is the Haar scaling function $\varphi(n) = 1$ for $n = 0, 1, \ldots, 7$ and 0 elsewhere, what is the resulting transform with respect to Haar wavelets?

**(b)** What is the transform if the input is the corresponding Haar wavelet function $\psi(n) = \{1, 1, 1, 1, -1, -1, -1, -1\}$ for $n = 0, 1, \ldots, 7$?

**(c)** What input sequence produces transform $\{0, 0, 0, 0, 0, 0, B, 0\}$ with nonzero coefficient $W_\psi(2, 2) = B$?

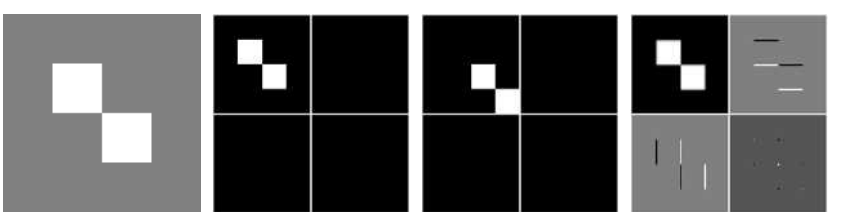
★**7.22** The two-dimensional fast wavelet transform is similar to the pyramidal coding scheme of Section 7.2.1. How are they similar? Given the three-scale wavelet transform in Fig. 7.10(a), how would you construct the corresponding approximation pyramid? How many levels would it have?

**7.23** Compute the two-dimensional wavelet transform with respect to Haar wavelets of the $2 \times 2$ image in Problem 7.9. Draw the required filter bank and label all inputs and outputs with the proper arrays.

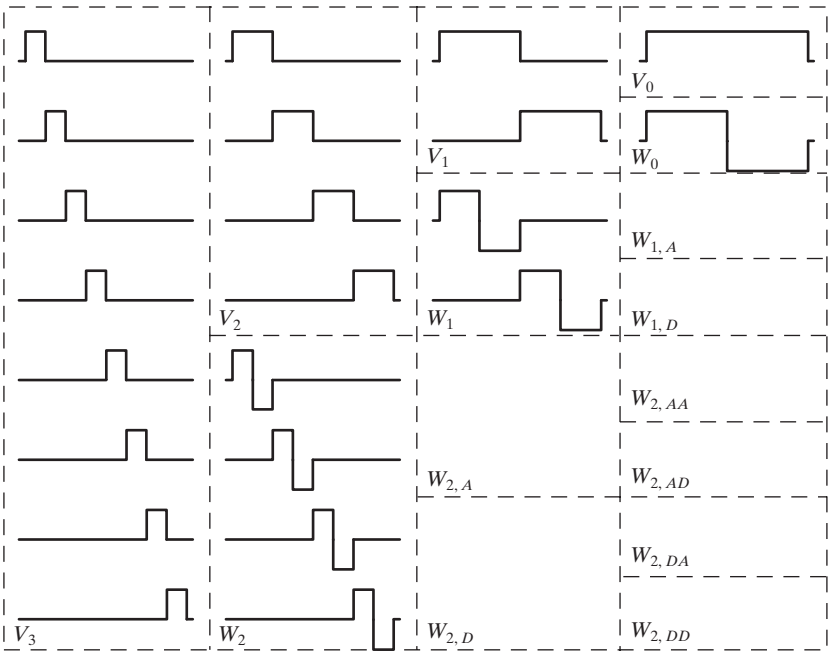
★**7.24** In the Fourier domain

$$f(x - x_0, y - y_0) \Leftrightarrow F(\mu, v)e^{-2\pi(\mu x_0/M + v y_0/N)}$$

and translation does not affect the display of $|F(\mu, v)|$. Using the following sequence of images, explain the translation property of wavelet transforms. The leftmost image contains two $32 \times 32$ white squares centered on a $128 \times 128$ gray background. The second image (from the left) is its single-scale wavelet transform with respect to Haar wavelets. The third is the wavelet transform of the original image after shifting it 32 pixels to the right and downward, and the final (rightmost) image is the wavelet transform of the original image after it has been shifted one pixel to the right and downward.



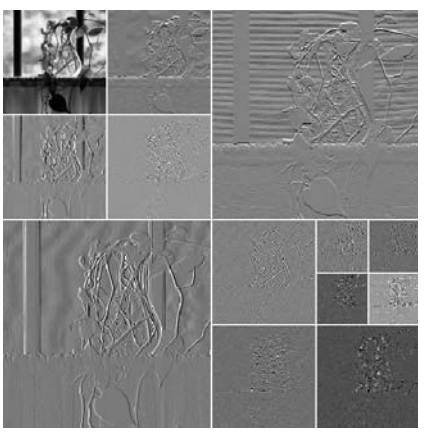
**7.25** The following table shows the Haar wavelet and scaling functions for a four-scale fast wavelet transform. Sketch the additional basis functions needed for a full three-scale packet decomposition. Give the mathematical expression or expressions for determining them. Then order the basis functions according to frequency content and explain the results.




**7.26** A wavelet packet decomposition of the vase from Fig. 7.1 is shown below.

**(a)** Draw the corresponding decomposition analysis tree, labeling all nodes with the names of the proper scaling and wavelet spaces.

**(b)** Draw and label the decomposition's frequency spectrum.



**7.27** Using the Haar wavelet, determine the minimum entropy packet decomposition for the function $f(n) = 0.25$ for $n = 0, 1, 2, \ldots, 15$. Employ the nonnormalized Shannon entropy,

$$E\big[f(n)\big] = \sum_n f^2(n) \ln\big[f^2(n)\big]$$

as the minimization criterion. Draw the optimal tree, labeling the nodes with the computed entropy values.