Chapter 2

Daubechies wavelets

> It is hardly an exaggeration to say that we will introduce almost as many analysis algorithms as there are signals... signals are so rich and complex that a single analysis method... cannot serve them all.
>
> *Yves Meyer*[1]

In this chapter we describe a large collection of wavelet transforms discovered by Daubechies. The Daubechies wavelet transforms are defined in the same way as the Haar wavelet transform—by computing running averages and differences via scalar products with scaling signals and wavelets—the only difference between them consists in how these scaling signals and wavelets are defined. For the Daubechies wavelet transforms, the scaling signals and wavelets have slightly longer supports, i.e., they produce averages and differences using just a few more values from the signal. This slight change, however, provides a tremendous improvement in the capabilities of these new transforms. They provide us with a set of powerful tools for performing basic signal processing tasks. These tasks include compression and noise removal for audio signals and for images, and include image enhancement and signal recognition.

2.1 The Daub4 wavelets

There are many Daubechies transforms, but they are all very similar. In this section we shall concentrate on the simplest one, the Daub4 wavelet

[1] Meyer's quote is from [ME2].

transform. The Daub4 wavelet transform is defined in essentially the same way as the Haar wavelet transform. If a signal $\mathbf{f}$ has an even number N of values, then the 1-level Daub4 transform is the mapping $\mathbf{f} \stackrel{\mathbf{D}_1}{\longmapsto} (\mathbf{a}^1 \,|\, \mathbf{d}^1)$ from the signal $\mathbf{f}$ to its first trend subsignal $\mathbf{a}^1$ and first fluctuation subsignal $\mathbf{d}^1$. Each value a_m of $\mathbf{a}^1 = (a_1, \ldots, a_{N/2})$ is equal to a scalar product:

$$a_m = \mathbf{f} \cdot \mathbf{V}_m^1 \tag{2.1}$$

of $\mathbf{f}$ with a 1-level *scaling signal* $\mathbf{V}_m^1$. Likewise, each value d_m of $\mathbf{d}^1 = (d_1, \ldots, d_{N/2})$ is equal to a scalar product:

$$d_m = \mathbf{f} \cdot \mathbf{W}_m^1 \tag{2.2}$$

of $\mathbf{f}$ with a 1-level *wavelet* $\mathbf{W}_m^1$. We shall define these Daub4 scaling signals and wavelets in a moment, but first we shall briefly describe the higher level Daub4 transforms.

The Daub4 wavelet transform, like the Haar transform, can be extended to multiple levels as many times as the signal length can be divided by 2. The extension is similar to the way the Haar transform is extended, i.e., by applying the 1-level Daub4 transform $\mathbf{D}_1$ to the first trend $\mathbf{a}^1$. This produces the mapping $\mathbf{a}^1 \stackrel{\mathbf{D}_1}{\longmapsto} (\mathbf{a}^2 \,|\, \mathbf{d}^2)$ from the first trend subsignal $\mathbf{a}^1$ to a second trend subsignal $\mathbf{a}^2$ and second fluctuation subsignal $\mathbf{d}^2$. The 2-level Daub4 transform $\mathbf{D}_2$ is then defined by the mapping $\mathbf{f} \stackrel{\mathbf{D}_2}{\longmapsto} (\mathbf{a}^2 \,|\, \mathbf{d}^2 \,|\, \mathbf{d}^1)$. For example, we show in Figure 2.2(b) the 2-level Daub4 transform of the signal shown in Figure 1.2(a). As with the Haar transform, the values of the second trend $\mathbf{a}^2$ and second fluctuation $\mathbf{d}^2$ can be obtained via scalar products with second-level scaling signals and wavelets. Likewise, the definition of a k-level Daub4 transform is obtained by applying the 1-level transform to the preceding level trend subsignal, just like in the Haar case. And, as in the Haar case, the values of the k-level trend subsignal $\mathbf{a}^k$ and fluctuation subsignal $\mathbf{d}^k$ are obtained as scalar products of the signal with k-level scaling signals and wavelets.

The difference between the Haar transform and the Daub4 transform lies in the way that the scaling signals and wavelets are defined. We shall first discuss the scaling signals. Let the *scaling numbers* $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ be defined by

$$\alpha_1 = \frac{1+\sqrt{3}}{4\sqrt{2}}, \quad \alpha_2 = \frac{3+\sqrt{3}}{4\sqrt{2}}, \quad \alpha_3 = \frac{3-\sqrt{3}}{4\sqrt{2}}, \quad \alpha_4 = \frac{1-\sqrt{3}}{4\sqrt{2}}. \tag{2.3}$$

Later in this chapter and the next, we shall describe how these scaling numbers were obtained. Using these scaling numbers, the 1-level Daub4 scaling signals are

$$
\begin{aligned}
\mathbf{V}_1^1 &= (\alpha_1, \alpha_2, \alpha_3, \alpha_4, 0, 0, \ldots, 0) \\
\mathbf{V}_2^1 &= (0, 0, \alpha_1, \alpha_2, \alpha_3, \alpha_4, 0, 0, \ldots, 0) \\
\mathbf{V}_3^1 &= (0, 0, 0, 0, \alpha_1, \alpha_2, \alpha_3, \alpha_4, 0, 0, \ldots, 0) \\
&\vdots \\
\mathbf{V}_{N/2-1}^1 &= (0, 0, \ldots, 0, \alpha_1, \alpha_2, \alpha_3, \alpha_4) \\
\mathbf{V}_{N/2}^1 &= (\alpha_3, \alpha_4, 0, 0, \ldots, 0, \alpha_1, \alpha_2).
\end{aligned} \tag{2.4}
$$

These scaling signals are all very similar to each other. For example, each scaling signal has a support of just four time-units. Notice also that the second scaling signal $\mathbf{V}_2^1$ is just a translation by two time-units of the first scaling signal $\mathbf{V}_1^1$. Likewise, the third scaling signal $\mathbf{V}_3^1$ is a translation by four time-units of $\mathbf{V}_1^1$, and each subsequent scaling signal is a translation by a multiple of two time-units of $\mathbf{V}_1^1$. There is one wrinkle here. For $\mathbf{V}_{N/2}^1$, we would have to translate $\mathbf{V}_1^1$ by $N-2$ time-units, but since $(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$ has length 4 this would send α_3 and α_4 beyond the length N of the signal $\mathbf{f}$. To avoid this problem, we *wrap-around* to the start; hence $\mathbf{V}_{N/2}^1 = (\alpha_3, \alpha_4, 0, 0, \ldots, 0, \alpha_1, \alpha_2)$. The Haar scaling signals also have this property of being translations by multiples of two time-units of the first scaling signal. But, since the first Haar scaling signal has a support of just two adjacent non-zero values, there is no wrap-around effect in the Haar case.

The second level Daub4 scaling signals are produced by repeating the operations that were used on the natural basis of signals $\mathbf{V}_1^0, \mathbf{V}_2^0, \ldots, \mathbf{V}_N^0$ to generate the first level scaling signals.[2] Using this natural basis, the first level Daub4 scaling signals satisfy

$$
\mathbf{V}_m^1 = \alpha_1 \mathbf{V}_{2m-1}^0 + \alpha_2 \mathbf{V}_{2m}^0 + \alpha_3 \mathbf{V}_{2m+1}^0 + \alpha_4 \mathbf{V}_{2m+2}^0 \tag{2.5a}
$$

with a wrap-around defined by $\mathbf{V}_{n+N}^0 = \mathbf{V}_n^0$. Similarly, the second level Daub4 scaling signals are defined by

$$
\mathbf{V}_m^2 = \alpha_1 \mathbf{V}_{2m-1}^1 + \alpha_2 \mathbf{V}_{2m}^1 + \alpha_3 \mathbf{V}_{2m+1}^1 + \alpha_4 \mathbf{V}_{2m+2}^1 \tag{2.5b}
$$

with a wrap-around defined by $\mathbf{V}_{n+N/2}^1 = \mathbf{V}_n^1$. Notice that this wrap-around, or periodicity, of the first level scaling signals is implied by the wrap-around invoked above for the natural signal basis.

By examining Formula (2.5b), we can see that each second-level Daub4 scaling signal, $\mathbf{V}_m^2$, lives for just 10 time-units, and is a translate by $4m$ time-units of $\mathbf{V}_1^2$ (if we include wrap-around). The second-level trend values are $\{\mathbf{f} \cdot \mathbf{V}_m^2\}$ and they measure trends over 10 successive values of $\mathbf{f}$, located

[2] This natural basis of signals was defined in (1.20).

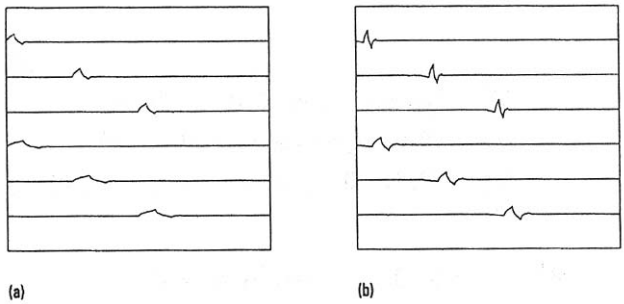

FIGURE 2.1
(a) The top 3 signals are 5-level Daub4 scaling signals $\mathbf{V}_1^5$, $\mathbf{V}_8^5$, and $\mathbf{V}_{16}^5$. The bottom three signals are 6-level scaling signals $\mathbf{V}_1^6$, $\mathbf{V}_4^6$, and $\mathbf{V}_8^6$. (b) The top 3 signals are 5-level Daub4 wavelets $\mathbf{W}_1^5$, $\mathbf{W}_8^5$, and $\mathbf{W}_{16}^5$. The bottom three signals are 6-level wavelets $\mathbf{W}_1^6$, $\mathbf{W}_4^6$, and $\mathbf{W}_8^6$.

in the same time positions as the non-zero values of $\mathbf{V}_m^2$. Hence these trends are measured over short time intervals that are shifts by multiples of 4 time-units of the interval consisting of the first 10 time-units. These 10-unit trends are slightly more than twice as long lasting as the trends measured by the first level scaling signals.

The k-level Daub4 scaling signals are defined by formulas similar to (2.5a) and (2.5b), but applied to the preceding level scaling signals. In Figure 2.1(a), we show some 5-level and 6-level Daub4 scaling signals. Notice that the supports of the 6-level scaling signals are about twice as long as the supports of the 5-level scaling signals. Figure 2.1(a) also illustrates the fact that each of the 5-level scaling signals is a translate of $\mathbf{V}_1^5$, and that each of the 6-level scaling signals is a translate of $\mathbf{V}_1^6$.

An important property of these scaling signals is that they all have energy 1. This is because of the following identity satisfied by the scaling numbers:

$$\alpha_1^2 + \alpha_2^2 + \alpha_3^2 + \alpha_4^2 = 1. \tag{2.6}$$

It is clear that (2.6) implies that each 1-level scaling signal has energy 1. To see that it also implies that each k-level scaling signal has energy 1 is more difficult; we will sketch the proof at the end of the next section.

Another identity satisfied by the scaling numbers is

$$\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 = \sqrt{2}. \tag{2.7}$$

This equation says that each 1-level trend value $\mathbf{f} \cdot \mathbf{V}_m^1$ is an average of four values of $\mathbf{f}$, multiplied by $\sqrt{2}$. It can also be shown that the sum of the ten successive non-zero values of $\mathbf{V}_m^2$ is 2, which shows that each 2-level trend value $\mathbf{f} \cdot \mathbf{V}_m^2$ is an average of ten successive values of $\mathbf{f}$, multiplied by 2.

Similarly, each k-level trend value $\mathbf{f} \cdot \mathbf{V}_m^k$ is an average of values of $\mathbf{f}$ over increasingly longer time intervals as k increases.

We now turn to a discussion of the Daub4 wavelets. Let the *wavelet numbers* $\beta_1, \beta_2, \beta_3, \beta_4$ be defined by

$$\beta_1 = \frac{1-\sqrt{3}}{4\sqrt{2}}, \quad \beta_2 = \frac{\sqrt{3}-3}{4\sqrt{2}}, \quad \beta_3 = \frac{3+\sqrt{3}}{4\sqrt{2}}, \quad \beta_4 = \frac{-1-\sqrt{3}}{4\sqrt{2}}. \tag{2.8}$$

Notice that the wavelet numbers are related to the scaling numbers by the equations: $\beta_1 = \alpha_4$, $\beta_2 = -\alpha_3$, $\beta_3 = \alpha_2$, and $\beta_4 = -\alpha_1$. Using these wavelet numbers, the 1-level Daub4 wavelets $\mathbf{W}_1^1, \ldots, \mathbf{W}_{N/2}^1$ are defined by

$$\begin{aligned}
\mathbf{W}_1^1 &= (\beta_1, \beta_2, \beta_3, \beta_4, 0, 0, \ldots, 0) \\
\mathbf{W}_2^1 &= (0, 0, \beta_1, \beta_2, \beta_3, \beta_4, 0, 0, \ldots, 0) \\
\mathbf{W}_3^1 &= (0, 0, 0, 0, \beta_1, \beta_2, \beta_3, \beta_4, 0, 0, \ldots, 0) \\
&\vdots \\
\mathbf{W}_{N/2-1}^1 &= (0, 0, \ldots, 0, \beta_1, \beta_2, \beta_3, \beta_4) \\
\mathbf{W}_{N/2}^1 &= (\beta_3, \beta_4, 0, 0, \ldots, 0, \beta_1, \beta_2).
\end{aligned} \tag{2.9}$$

These wavelets are all translates of $\mathbf{W}_1^1$, with a wrap-around for the last wavelet. Each wavelet has a support of just four time-units, corresponding to the four non-zero wavelet numbers used to define them. The 1-level Daub4 wavelets satisfy

$$\mathbf{W}_m^1 = \beta_1 \mathbf{V}_{2m-1}^0 + \beta_2 \mathbf{V}_{2m}^0 + \beta_3 \mathbf{V}_{2m+1}^0 + \beta_4 \mathbf{V}_{2m+2}^0.$$

Similarly, the 2-level Daub4 wavelets are defined by

$$\mathbf{W}_m^2 = \beta_1 \mathbf{V}_{2m-1}^1 + \beta_2 \mathbf{V}_{2m}^1 + \beta_3 \mathbf{V}_{2m+1}^1 + \beta_4 \mathbf{V}_{2m+2}^1.$$

All other levels of Daub4 wavelets are defined in a similar fashion. In Figure 2.1(b) we show some of the Daub4 wavelets. It is interesting to compare them with the Daub4 scaling functions shown in Figure 2.1(a).

The Daub4 wavelets all have energy 1. This is clear for the 1-level Daub4 wavelets, since

$$\beta_1^2 + \beta_2^2 + \beta_3^2 + \beta_4^2 = 1. \tag{2.10}$$

It can also be shown that all k-level Daub4 wavelets have energy 1 as well.

Each fluctuation value $d_m = \mathbf{f} \cdot \mathbf{W}_m^1$ can be viewed as a differencing operation on the values of $\mathbf{f}$ because

$$\beta_1 + \beta_2 + \beta_3 + \beta_4 = 0. \tag{2.11}$$

Equation (2.11) is a generalization of the Haar case, where we had $1/\sqrt{2} - 1/\sqrt{2} = 0$. It also implies, as with the Haar case, that a fluctuation value

$\mathbf{f} \cdot \mathbf{W}_m^1$ will be zero if the signal $\mathbf{f}$ is constant over the support of a Daub4 wavelet $\mathbf{W}_m^1$. Much more is true, however. Not only is (2.11) true, but we also have

$$0\beta_1 + 1\beta_2 + 2\beta_3 + 3\beta_4 = 0. \tag{2.12}$$

Equations (2.11) and (2.12), and Equation (2.7), imply the following property for the k-level Daub4 wavelet transform.

Property I. *If a signal* $\mathbf{f}$ *is (approximately) linear over the support of a* k*-level Daub4 wavelet* $\mathbf{W}_m^k$*, then the* k*-level fluctuation value* $\mathbf{f} \cdot \mathbf{W}_m^k$ *is (approximately) zero.*

For the 1-level case, Property I follows easily from Equations (2.11) and (2.12). It is more difficult to prove Property I for the k-level case, when $k > 1$, and it is for such cases that Equation (2.7) is needed.

To see why Property I is so important, we examine how it relates to sampled signals. In Figure 2.2(a) we show a signal obtained from uniformly spaced samples over the interval $[0, 1)$ of a function which has a continuous second derivative. As shown in Figure 2.2(b), the 1-level and 2-level fluctuations $\mathbf{d}^1$ and $\mathbf{d}^2$ have values that are all very near zero. This is because a large proportion of the signal consists of values that are approximately linear over a support of one of the Daub4 wavelets. For example, in Figures 2.2(c) and (d) we show magnifications of small squares centered at the points $(.296, .062)$ and $(.534, .067)$. It is clear from these figures that the signal values are approximately linear within these small squares. This is true of a large number of points on the graph of the signal and implies that many of the fluctuation values for this signal will be near zero. The basic principles of Calculus tell us that this example is typical for a signal that is sampled from a function that has a continuous second derivative. We shall discuss this point in more detail at the end of this section.

Each level Daub4 transform has an inverse. The inverse for the 1-level Daub4 transform, which maps the transform $(\mathbf{a}^1 \,|\, \mathbf{d}^1)$ back to the signal $\mathbf{f}$, is calculated explicitly as

$$\mathbf{f} = \mathbf{A}^1 + \mathbf{D}^1 \tag{2.13a}$$

with *first averaged signal* $\mathbf{A}^1$ defined by

$$\begin{aligned} \mathbf{A}^1 &= a_1\mathbf{V}_1^1 + a_2\mathbf{V}_2^1 + \cdots + a_{N/2}\mathbf{V}_{N/2}^1 \\ &= (\mathbf{f} \cdot \mathbf{V}_1^1)\mathbf{V}_1^1 + (\mathbf{f} \cdot \mathbf{V}_2^1)\mathbf{V}_2^1 + \cdots + (\mathbf{f} \cdot \mathbf{V}_{N/2}^1)\mathbf{V}_{N/2}^1 \end{aligned} \tag{2.13b}$$

and *first detail signal* $\mathbf{D}^1$ defined by

$$\begin{aligned} \mathbf{D}^1 &= d_1\mathbf{W}_1^1 + d_2\mathbf{W}_2^1 + \cdots + d_{N/2}\mathbf{W}_{N/2}^1 \\ &= (\mathbf{f} \cdot \mathbf{W}_1^1)\mathbf{W}_1^1 + (\mathbf{f} \cdot \mathbf{W}_2^1)\mathbf{W}_2^1 + \cdots + (\mathbf{f} \cdot \mathbf{W}_{N/2}^1)\mathbf{W}_{N/2}^1. \end{aligned} \tag{2.13c}$$

Formulas (2.13a) through (2.13c) are generalizations of similar formulas that we found for the Haar transform [see Section 1.4]. They are the first stage in a Daub4 MRA of the signal $\mathbf{f}$. We will not take the time at this point to prove that these formulas are correct; their proof involves techniques from the field of linear algebra. For those readers who are familiar with linear algebra, we provide a proof at the end of the next section. It is more important now to reflect on what these formulas mean.

Formula (2.13a) shows that the signal $\mathbf{f}$ can be expressed as a sum of an averaged signal $\mathbf{A}^1$ plus a detail signal $\mathbf{D}^1$. Because of Formula (2.13b) we can see that the averaged signal $\mathbf{A}^1$ is a combination of elementary scaling signals. Each scaling signal $\mathbf{V}_m^1$ is a short-lived signal, whose support consists of just four consecutive time-indices; the relative contribution of each scaling signal to $\mathbf{A}^1$ is measured by the trend value $a_m = \mathbf{f} \cdot \mathbf{V}_m^1$. Thus $\mathbf{A}^1$ is a sum of short-lived components which are multiples of the scaling signals $\mathbf{V}_m^1$. These scaling signals move across the time-axis in steps of just two time-units and live for only four time-units; *they measure short-lived trends in the signal via the trend values* $a_m = \mathbf{f} \cdot \mathbf{V}_m^1$. Likewise, the detail signal $\mathbf{D}^1$ is a combination of elementary wavelets $\mathbf{W}_m^1$. These wavelets $\mathbf{W}_m^1$ march across the time-axis in steps of two time-units and live for only four time-units. The relative contribution of each wavelet to $\mathbf{D}^1$ is measured by the fluctuation value $d_m = \mathbf{f} \cdot \mathbf{W}_m^1$. Since $\mathbf{D}^1 = \mathbf{f} - \mathbf{A}^1$, the sum of all of these short-lived fluctuations $\{d_m \mathbf{W}_m^1\}$ equals the difference between the signal $\mathbf{f}$ and its lower resolution, averaged, version $\mathbf{A}^1$. Because the 1-level wavelets $\{\mathbf{W}_m^1\}$ live for only four time-units and march across the time-axis in steps of two units, *they are able to detect very short-lived, transient, fluctuations in the signal.*

The inverse of the 2-level Daub4 transform is described by the formula

$$\mathbf{f} = \mathbf{A}^2 + \mathbf{D}^2 + \mathbf{D}^1 \tag{2.14a}$$

where

$$\mathbf{A}^2 = (\mathbf{f} \cdot \mathbf{V}_1^2)\mathbf{V}_1^2 + \cdots + (\mathbf{f} \cdot \mathbf{V}_{N/4}^2)\mathbf{V}_{N/4}^2$$

$$\mathbf{D}^2 = (\mathbf{f} \cdot \mathbf{W}_1^2)\mathbf{W}_1^2 + \cdots + (\mathbf{f} \cdot \mathbf{W}_{N/4}^2)\mathbf{W}_{N/4}^2 \tag{2.14b}$$

are the second averaged signal and second detail signal, respectively. The signal $\mathbf{D}^1$ is the first detail signal which we defined above. The second averaged signal $\mathbf{A}^2$ is a sum of components which are multiples of the scaling signals $\mathbf{V}_m^2$; these components move across the time-axis in steps of four time-units and live for ten time-units. The relative contribution of each scaling signal $\mathbf{V}_m^2$ to $\mathbf{A}^2$ is measured by the trend value $\mathbf{f} \cdot \mathbf{V}_m^2$. Since $\mathbf{A}^1 = \mathbf{A}^2 + \mathbf{D}^2$, the second detail signal $\mathbf{D}^2$ provides the details needed to produce the first averaged signal from the second averaged signal. This second detail signal $\mathbf{D}^2$ is a combination of the wavelets $\mathbf{W}_m^2$, which move

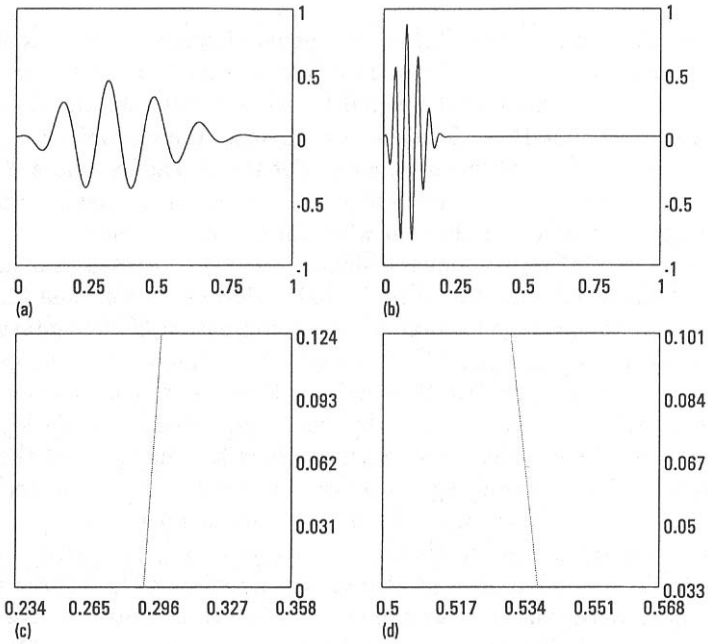

FIGURE 2.2
(a) Signal. (b) 2-level Daub4 transform. The trend a^2 is graphed over [0, .25), while the fluctuations d^2 and d^1 are graphed over [.25, .5) and [.5, 1), respectively. (c) and (d) Magnifications of the Signal's graph in two small squares; the Signal is approximately linear.

across the time-axis in steps of four time-units and live for ten time-units. The relative contribution of each wavelet $\mathbf{W}_m^2$ to $\mathbf{D}^2$ is measured by the fluctuation value $\mathbf{f} \cdot \mathbf{W}_m^2$. Like the 1-level wavelets, the 2-level wavelets are also able to detect transient fluctuations in a signal, but their supports are 10 units long instead of 4 units long. Hence the scale on which the 2-level wavelets measure fluctuations is slightly more than twice as long as the scale on which the 1-level wavelets measure fluctuations.

Further levels of the Daub4 transform are handled in a like manner. The k-level Daub4 transform has an inverse that produces the following MRA of the signal $\mathbf{f}$:

$$\mathbf{f} = \mathbf{A}^k + \mathbf{D}^k + \cdots + \mathbf{D}^2 + \mathbf{D}^1.$$

The formulas for $\mathbf{A}^k$ and $\mathbf{D}^k$ are (for $N_k = N/2^k$):

$$\mathbf{A}^k = (\mathbf{f} \cdot \mathbf{V}_1^k)\mathbf{V}_1^k + \cdots + (\mathbf{f} \cdot \mathbf{V}_{N_k}^k)\mathbf{V}_{N_k}^k$$

and

$$\mathbf{D}^k = (\mathbf{f} \cdot \mathbf{W}_1^k)\mathbf{W}_1^k + \cdots + (\mathbf{f} \cdot \mathbf{W}_{N_k}^k)\mathbf{W}_{N_k}^k.$$

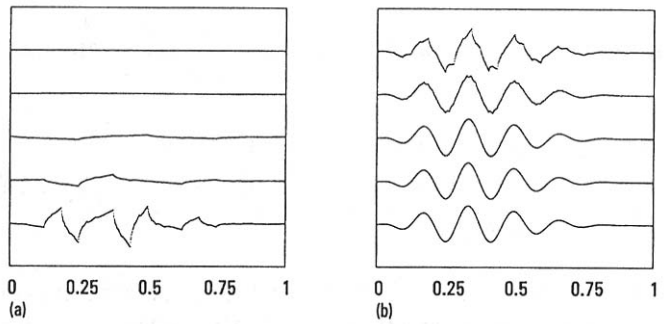

FIGURE 2.3
Daub4 MRA of the signal shown in Figure 1.1(a). The graphs are of the 10 averaged signals $\mathbf{A}^{10}$ through $\mathbf{A}^{1}$. Beginning with $\mathbf{A}^{10}$ on the top left down to $\mathbf{A}^{6}$ on the bottom left, then $\mathbf{A}^{5}$ on the top right down to $\mathbf{A}^{1}$ on the bottom right. Compare with Figure 1.3.

In Figure 2.3 we show a Daub4 MRA of the same signal that we analyzed in Chapter 1 using a Haar MRA (see Figure 1.3). It is interesting to compare these two MRAs. The Daub4 MRA appears to be the superior one; it converges more quickly towards the original signal. The Daub4 averaged signals $\mathbf{A}^3$ through $\mathbf{A}^1$ all appear to be equally close approximations of the original signal. This is due to the fact that the values of the first and second Daub4 fluctuation subsignals $\mathbf{d}^1$ and $\mathbf{d}^2$ are so small [see Figure 2.2(b)] that they can be neglected without losing much detail. Likewise, the third Daub4 fluctuation subsignal $\mathbf{d}^3$ has negligible values. The corresponding Daub4 detail signals $\mathbf{D}^1$, $\mathbf{D}^2$, and $\mathbf{D}^3$ contribute very little detail to the signal; hence $\mathbf{f} \approx \mathbf{A}^3$ is a very good approximation. Another advantage of the Daub4 MRA is that the jumpy, or clumpy, appearance of the Haar averaged signals does not appear in the Daub4 averaged signals.

Further remarks on Property I*

In discussing Property I above, we showed by means of an example that it applies to sampled signals when the analog signal has a continuous second derivative over the support of a Daub4 wavelet. That is, we assume that the signal $\mathbf{f}$ has values satisfying $f_n = g(t_n)$ for $n = 1, 2, \ldots, N$, and that the function g has a continuous second derivative over the support of a Daub4 wavelet. For simplicity we shall assume that this is a 1-level wavelet, say $\mathbf{W}_m^1$. We can then write

$$g(t_{2m-1+k}) = g(t_{2m-1}) + g'(t_{2m-1})(kh) + \mathrm{O}(h^2) \tag{2.15}$$

where $\mathrm{O}(h^2)$ stands for a quantity that is a bounded multiple of h^2. The number h is the constant step-size $h = t_{n+1} - t_n$, which holds for each n.

Making use of Equation (2.15), and Equations (2.11) and (2.12), we find that

$$\begin{aligned}\mathbf{f} \cdot \mathbf{W}_m^1 &= g(t_{2m-1})\{\beta_1 + \beta_2 + \beta_3 + \beta_4\} \\ &\quad + g'(t_{2m-1})h\{0\beta_1 + 1\beta_2 + 2\beta_3 + 3\beta_4\} + \mathrm{O}(h^2) \\ &= \mathrm{O}(h^2).\end{aligned}$$

Thus $\mathbf{f} \cdot \mathbf{W}_m^1 = \mathrm{O}(h^2)$. This illustrates Property I, since h is generally much smaller than 1 and consequently h^2 is very tiny indeed. Our discussion also shows why the Daub4 transform generally produces much smaller fluctuation values than the Haar transform does, since for the Haar transform it is typically possible only to have $\mathbf{f} \cdot \mathbf{W}_m^1 = \mathrm{O}(h)$, which is generally much larger than $\mathrm{O}(h^2)$.

2.2 Conservation and compaction of energy

Like the Haar transform, a Daubechies wavelet transform conserves the energy of signals and redistributes this energy into a more compact form. In this section we shall discuss these properties as they relate to the Daub4 transform, but the general principles apply to all of the various Daubechies transforms.

Let's begin with a couple of examples. First, consider the signal $\mathbf{f}$ graphed in Figure 2.2(a). Using FAWAV we calculate that its energy is 509.2394777. On the other hand, the energy of its 1-level Daub4 transform is also 509.2394777. This illustrates the conservation of energy property of the Daub4 transform. Since the 2-level Daub4 transform consists of applying the 1-level Daub4 transform to the first trend subsignal, it follows that the 2-level Daub4 transform also conserves the energy of the signal $\mathbf{f}$. Likewise, a k-level Daub4 transform conserves energy as well.

As with the Haar transform, the Daub4 transform also redistributes the energy of the signal into a more compact form. For example, consider the signals shown in Figure 2.4. In Figure 2.4(b) we show the 2-level Daub4 transform of the signal graphed in Figure 1.2(a). Its cumulative energy profile is graphed in Figure 2.4(d). This cumulative energy profile shows that the 2-level Daub4 transform has redistributed almost all of the energy of the signal into the second trend subsignal, which is graphed over the first quarter of the time-interval. For comparison, we also show in Figures 2.4(a) and (c) the 2-level Haar transform and its cumulative energy profile. It is obvious from these graphs that the Daub4 transform achieves a more compact redistribution of the energy of the signal.

Justification of conservation of energy *

We shall now show why the Daub4 transform preserves the energy of each signal, and provide justifications for a couple of other statements made in the preceding section. Readers who are not familiar with linear algebra, especially matrix algebra, should skip this discussion. It will not play a major role in the material that follows, which will stress the applications of the Daubechies transforms.

To begin, define the matrix $\mathcal{D}_N$ by

$$\mathcal{D}_N = \begin{pmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ \beta_1 & \beta_2 & \beta_3 & \beta_4 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & \beta_1 & \beta_2 & \beta_3 & \beta_4 & 0 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots \\ \alpha_3 & \alpha_4 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & \alpha_1 & \alpha_2 \\ \beta_3 & \beta_4 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & \beta_1 & \beta_2 \end{pmatrix}. \tag{2.16}$$

Notice that the rows of $\mathcal{D}_N$ are the first-level Daub4 scaling signals and wavelets. These scaling signals and wavelets satisfy

$$\mathbf{V}_n^1 \cdot \mathbf{V}_m^1 = \begin{cases} 1 & \text{if } n = m \\ 0 & \text{if } n \neq m, \end{cases} \tag{2.17a}$$

$$\mathbf{W}_n^1 \cdot \mathbf{W}_m^1 = \begin{cases} 1 & \text{if } n = m \\ 0 & \text{if } n \neq m, \end{cases} \tag{2.17b}$$

$$\mathbf{V}_n^1 \cdot \mathbf{W}_m^1 = 0 \quad \text{all } m, n. \tag{2.17c}$$

These equations show that the rows of $\mathcal{D}_N$ form an orthonormal set of vectors, i.e., that $\mathcal{D}_N$ is an orthogonal matrix. Another way of expressing these equations is

$$\mathcal{D}_N^{\mathrm{T}} \mathcal{D}_N = I_N \tag{2.18}$$

where $\mathcal{D}_N^{\mathrm{T}}$ is the transpose of the matrix $\mathcal{D}_N$ and I_N is the N by N identity matrix.

We can now show that the Daub4 transform preserves the energy of a signal $\mathbf{f}$. *These arguments will only make use of Equations (2.17a) to (2.17c), and (2.18). Therefore they will apply to all of the Daubechies transforms described in this chapter, since all of the Daubechies scaling signals and wavelets will satisfy these same equations.* The matrix $\mathcal{D}_N$ will, in each case, be defined by rows consisting of the 1-level scaling signals and wavelets.

Comparing the definition of the matrix $\mathcal{D}_N$ and the definition of the 1-level Daub4 transform, we see that

$$(a_1, d_1, a_2, d_2, \dots, a_{N/2}, d_{N/2})^{\mathrm{T}} = \mathcal{D}_N \, \mathbf{f}^{\mathrm{T}}.$$

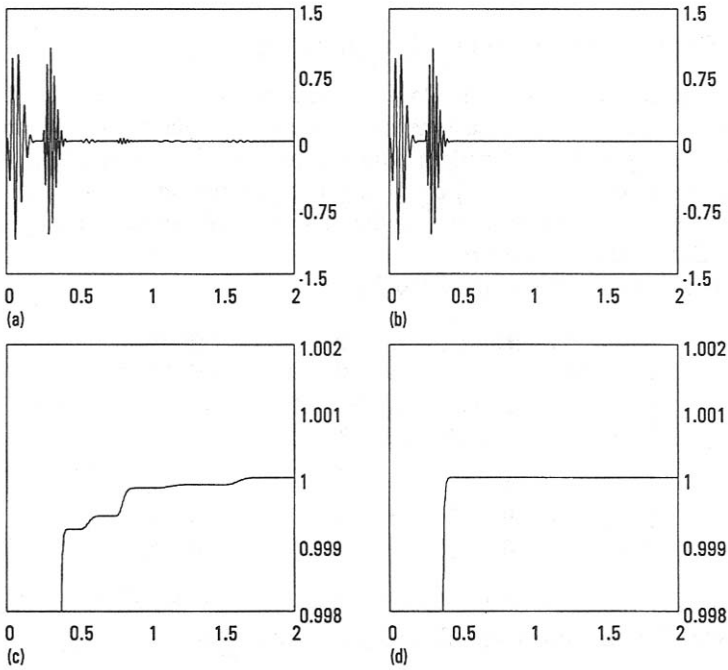

FIGURE 2.4
(a) Graph of 2-level Haar transform of signal in Figure 1.2(a). (b) Graph of 2-level Daub4 transform of same signal. (c) and (d) Cumulative energy profiles of the transforms in (a) and (b), respectively.

Therefore,

$$a_1^2 + d_1^2 + \cdots + a_{N/2}^2 + d_{N/2}^2 = \left(\mathcal{D}_N\,\mathbf{f}^{\mathrm{T}}\right)^{\mathrm{T}}\left(\mathcal{D}_N\,\mathbf{f}^{\mathrm{T}}\right).$$

Furthermore, the energy $\mathcal{E}_{(\mathbf{a}^1\,|\,\mathbf{d}^1)}$ of the 1-level Daub4 transform of $\mathbf{f}$ satisfies

$$a_1^2 + \cdots + a_{N/2}^2 + d_1^2 + \cdots + d_{N/2}^2 = \mathcal{E}_{(\mathbf{a}^1\,|\,\mathbf{d}^1)}.$$

Since the left-hand sides of these last two equations are clearly equal, we make use of (2.18) to obtain

$$\begin{aligned}\mathcal{E}_{(\mathbf{a}^1\,|\,\mathbf{d}^1)} &= (\mathcal{D}_N\mathbf{f}^{\mathrm{T}})^{\mathrm{T}}\,(\mathcal{D}_N\mathbf{f}^{\mathrm{T}})\\ &= \mathbf{f}\,\mathcal{D}_N^{\mathrm{T}}\,\mathcal{D}_N\,\mathbf{f}^{\mathrm{T}}\\ &= \mathbf{f}\,\mathbf{f}^{\mathrm{T}}\\ &= \mathcal{E}_{\mathbf{f}}.\end{aligned}$$

This proves that the 1-level Daub4 transform has the Conservation of Energy property. As we argued above, this also shows that every level Daub4 transform conserves energy.

Another consequence of Equations (2.17a) to (2.17c) is that the Daub4 scaling signals and wavelets all have energy 1. Since the energy $\mathcal{E}_{\mathbf{f}}$ of a signal equals $\mathbf{f} \cdot \mathbf{f}$, these equations show immediately that $\mathbf{V}_m^1$ and $\mathbf{W}_m^1$ each have energy 1. To indicate why all other scaling signals and wavelets have energy 1, we will show why the wavelet $\mathbf{W}_1^2$ has energy 1. Similar arguments can be used for the other wavelets and scaling signals. Since

$$\mathbf{W}_1^2 = \beta_1 \mathbf{V}_1^1 + \beta_2 \mathbf{V}_2^1 + \beta_3 \mathbf{V}_3^1 + \beta_4 \mathbf{V}_4^1$$

we find, using (2.17a), that

$$\mathbf{W}_1^2 \cdot \mathbf{W}_1^2 = \beta_1^2 \mathbf{V}_1^1 \cdot \mathbf{V}_1^1 + \beta_2^2 \mathbf{V}_2^1 \cdot \mathbf{V}_2^1 + \beta_3^2 \mathbf{V}_3^1 \cdot \mathbf{V}_3^1 + \beta_4^2 \mathbf{V}_4^1 \cdot \mathbf{V}_4^1. \quad (2.19)$$

Notice that terms such as $\beta_n \beta_m \mathbf{V}_n^1 \cdot \mathbf{V}_m^1$ are equal to 0 when $m \neq n$; so they do not appear on the right-hand side of (2.19). Each scalar product on the right-hand side of (2.19) is 1; hence

$$\mathbf{W}_1^2 \cdot \mathbf{W}_1^2 = \beta_1^2 + \beta_2^2 + \beta_3^2 + \beta_4^2 = 1$$

and that proves that the energy of $\mathbf{W}_1^2$ is 1.

Similar arguments can be used to show that Equations (2.13a) through (2.13c) are valid. We shall briefly indicate why this is so. Suppose that a signal $\mathbf{f}$ is defined by

$$r_1 \mathbf{V}_1^1 + \cdots + r_{N/2} \mathbf{V}_{N/2}^1 + s_1 \mathbf{W}_1^1 + \cdots + s_{N/2} \mathbf{W}_{N/2}^1 = \mathbf{f} \quad (2.20)$$

where $r_1, \ldots, r_{N/2}, s_1, \ldots, s_{N/2}$ are constants. Then, by using Equations (2.17a) through (2.17c), it follows that

$$r_m = \mathbf{f} \cdot \mathbf{V}_m^1, \quad s_m = \mathbf{f} \cdot \mathbf{W}_m^1 \quad (2.21)$$

for each m. In particular, if $\mathbf{f} = (0, 0, \ldots, 0)$, then $r_1 = 0, r_2 = 0, \ldots, r_{N/2} = 0$ and $s_1 = 0, s_2 = 0, \ldots, s_{N/2} = 0$. This proves that the signals

$$\mathbf{V}_1^1, \ldots, \mathbf{V}_{N/2}^1, \mathbf{W}_1^1, \ldots, \mathbf{W}_{N/2}^1$$

are linearly independent; hence they form a basis for the vector space $\mathbf{R}^N$ of all real-valued signals of length N. Consequently Equation (2.20) must hold, with unique coefficients $r_1, \ldots, r_{N/2}, s_1, \ldots, s_{N/2}$, for every signal $\mathbf{f}$. And Formula (2.21) shows that these coefficients are equal to the scalar products of $\mathbf{f}$ with the scaling signals and wavelets, exactly as described in Equations (2.13a) through (2.13c).

How wavelet and scaling numbers are found *

In this subsection we shall briefly outline how the Daub4 scaling numbers and wavelet numbers are determined. The essential features of this outline

also apply to the other Daubechies wavelets that are defined in the next section.

The constraints that determine the Daub4 scaling and wavelet numbers are Equations (2.17a) to (2.17c), (2.6), (2.7), and (2.10) through (2.12). By combining these last three equations with the requirement that $\mathbf{W}_1^1 \cdot \mathbf{W}_2^1 = 0$ from (2.17b), we obtain the following four constraining equations on the wavelet numbers:

$$\begin{aligned} \beta_1^2 + \beta_2^2 + \beta_3^2 + \beta_4^2 &= 1 \\ \beta_1 + \beta_2 + \beta_3 + \beta_4 &= 0 \\ 0\beta_1 + 1\beta_2 + 2\beta_3 + 3\beta_4 &= 0 \\ \beta_1\beta_3 + \beta_2\beta_4 &= 0. \end{aligned}$$

These equations are sufficient for uniquely determining (except for multiplication by -1) the values for β_1, β_2, β_3, and β_4 for the Daub4 wavelet numbers. The scaling numbers are then determined by the equations $\alpha_1 = -\beta_4$, $\alpha_2 = \beta_3$, $\alpha_3 = -\beta_2$, and $\alpha_4 = \beta_1$.

This very brief outline only partially answers the question of how scaling numbers and wavelet numbers are found. We shall provide a more complete discussion in the next chapter.

2.3 Other Daubechies wavelets

In this section we shall complete our introduction to the theory of the Daubechies wavelets and wavelet transforms. In the previous two sections we described the Daub4 wavelet transform and its associated set of scaling signals and wavelets. We shall now complete our discussion by describing the various Daub$J$ transforms for $J = 6, 8, \ldots, 20$, and by describing the Coif$I$ transforms for $I = 6, 12, 18, 24, 30$. These wavelet transforms are all quite similar to the Daub4 transform; our treatment here will concentrate on the value of having more wavelet transforms at our disposal. There are also many more wavelet transforms—such as *spline wavelet transforms,* various types of *biorthogonal wavelet transforms,* and even more Daub$J$ and Coif$I$ transforms than the ones we describe—but we shall not try to give an exhaustive coverage of all of these transforms. Examining a few trees should give us a good feeling for the forest of wavelet transforms.

Let's begin with the Daub$J$ transforms for $J = 6, 8, \ldots, 20$. The easiest way to understand these transforms is just to treat them as simple generalizations of the Daub4 transform. The most obvious difference between them is the length of the supports of their scaling signals and wavelets. For

example, for the Daub6 wavelet transform, we define the scaling numbers $\alpha_1, \ldots, \alpha_6$ to be

$$\begin{aligned} \alpha_1 &= 0.332670552950083, & \alpha_2 &= 0.806891509311092, \\ \alpha_3 &= 0.459877502118491, & \alpha_4 &= -0.135011020010255, \\ \alpha_5 &= -0.0854412738820267, & \alpha_6 &= 0.0352262918857095 \end{aligned}$$

and the wavelet numbers $\beta_1, \ldots, \beta_6$ to be

$$\beta_1 = \alpha_6,\ \beta_2 = -\alpha_5,\ \beta_3 = \alpha_4,\ \beta_4 = -\alpha_3,\ \beta_5 = \alpha_2,\ \beta_6 = -\alpha_1.$$

We then generalize the formulas in (2.4) in the following way:

$$\begin{aligned} \mathbf{V}_1^1 &= (\alpha_1, \alpha_2, \alpha_3\, a_4, \alpha_5, \alpha_6, 0, 0, \ldots, 0) \\ \mathbf{V}_2^1 &= (0, 0, \alpha_1, \alpha_2, \alpha_3\, a_4, \alpha_5, \alpha_6, 0, 0, \ldots, 0) \\ \mathbf{V}_3^1 &= (0, 0, 0, 0, \alpha_1, \alpha_2, \alpha_3\, a_4, \alpha_5, \alpha_6, 0, 0, \ldots, 0) \\ &\vdots \\ \mathbf{V}_{N/2}^1 &= (\alpha_3\, a_4, \alpha_5, \alpha_6, 0, 0, \ldots, 0, \alpha_1, a_2) \end{aligned} \tag{2.22}$$

with a wrap-around occurring for $\mathbf{V}_{N/2-1}^1$ and $\mathbf{V}_{N/2}^1$. The formulas in (2.22) define the first level Daub6 scaling signals. The scaling numbers satisfy (to a high degree of accuracy):

$$\alpha_1^2 + \alpha_2^2 + \alpha_3^2 + \alpha_4^2 + \alpha_5^2 + \alpha_6^2 = 1, \tag{2.23a}$$

$$\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 + \alpha_5 + \alpha_6 = \sqrt{2}. \tag{2.23b}$$

Equation (2.23a) says that each scaling signal $\mathbf{V}_m^1$ has an energy of 1, while Equation (2.23b) says that the trend values $\mathbf{f} \cdot \mathbf{V}_m^1$ are averages of six successive values of $\mathbf{f}$, multiplied by $\sqrt{2}$.

The 1-level Daub6 wavelets are defined via the wavelet numbers $\beta_1, \ldots, \beta_6$ in the same manner. In fact, since all of the definitions and formulas given in the last two sections generalize in obvious ways, we shall not repeat them.

Let's consider instead what new features are exhibited by the Daub6 transform. The principal feature is that the wavelet numbers $\beta_1, \ldots, \beta_6$ satisfy the following three identities (to a high degree of accuracy):

$$\begin{aligned} \beta_1 + \beta_2 + \beta_3 + \beta_4 + \beta_5 + \beta_6 &= 0, \\ 0\beta_1 + 1\beta_2 + 2\beta_3 + 3\beta_4 + 4\beta_5 + 5\beta_6 &= 0, \\ 0^2\beta_1 + 1^2\beta_2 + 2^2\beta_3 + 3^2\beta_4 + 4^2\beta_5 + 5^2\beta_6 &= 0. \end{aligned} \tag{2.24}$$

These equations, along with Equation (2.23b), imply the following property.

Property II. *If a signal* $\mathbf{f}$ *is (approximately) quadratic over the support of a k-level Daub6 wavelet* $\mathbf{W}_m^k$*, then the k-level Daub6 fluctuation value* $\mathbf{f} \cdot \mathbf{W}_m^k$ *is (approximately) zero.*

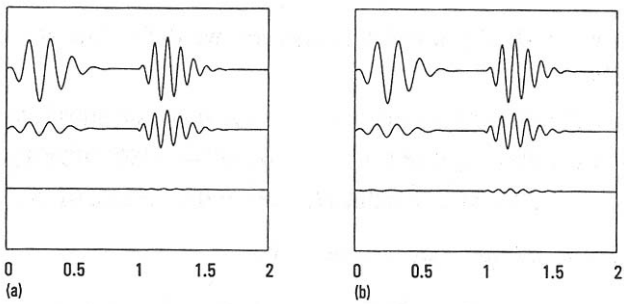

FIGURE 2.5
(a) Top: Signal. Middle: 1-level Daub4 fluctuation subsignal (multiplied by 1000 for comparison with the signal). Bottom: 1-level Daub6 fluctuation subsignal (also multiplied by 1000). (b) Similar graphs for 3-level Daub4 and Daub6 fluctuation subsignals (multiplied by 30).

Because of this property, the Daub6 transform will often produce smaller size fluctuation values than those produced by the Daub4 transform. The types of signals for which this occurs are the ones that are obtained from samples of analog signals that are at least three times continuously differentiable (at least over large portions of the analog signal). These kinds of signals are better approximated, over a large proportion of their values, by quadratic approximations rather than just linear approximations. Quadratic functions have curved graphs and can thereby provide superior approximations to the parts of the signal that are near to the turning points in its graph. To illustrate these ideas, consider the signal graphed at the top of Figure 2.5(a) and its 1-level Daub4 and Daub6 fluctuation subsignals graphed in the middle and at the bottom of the figure, respectively. This figure makes it clear that the Daub4 fluctuation values are significantly larger in magnitude than the Daub6 fluctuation values. It also shows that the largest magnitude Daub4 fluctuation values occur near the turning points in the graph of the signal. Similar graphs in Figure 2.5(b) illustrate the same ideas for the 3-level Daub4 and Daub6 fluctuation values.

When the goal is compression of signals, such as musical tones which often have graphs like the signal at the top of Figure 2.5(a), then the Daub6 transform can generally perform better at compressing the signal than the Daub4 transform. This is due to the larger number of Daub6 fluctuation values which can be ignored as insignificant. When the goal, however, is identifying features of the signal that are related to turning points in its graph, then the Daub4 transform can identify the location of these turning points more clearly as shown in Figure 2.5.

The other Daubechies wavelet transforms, the Daub$J$ transforms for $J = 8, 10, \ldots, 20$, are defined in essentially the same way. The scaling numbers

$\alpha_1, \ldots, \alpha_J$ satisfy

$$\alpha_1^2 + \alpha_2^2 + \cdots + \alpha_J^2 = 1, \tag{2.25a}$$

$$\alpha_1 + \alpha_2 + \cdots + \alpha_J = \sqrt{2}. \tag{2.25b}$$

And the wavelet numbers $\beta_1, \ldots, \beta_J$ are defined by

$$\beta_1 = \alpha_J,\ \beta_2 = -\alpha_{J-1},\ \beta_3 = \alpha_{J-2}, \ldots,\ \beta_{J-1} = \alpha_2,\ \beta_J = -\alpha_1. \tag{2.26}$$

These wavelet numbers satisfy the following identities (we set $0^0 = 1$ to enable a single statement):

$$0^L\beta_1 + 1^L\beta_2 + \ldots + (J-1)^L\beta_J = 0, \quad \text{for } L = 0, 1, \ldots, J/2 - 1. \tag{2.27}$$

These identities, along with (2.25b), imply the following property which is a generalization of Properties I and II above.

Property III. *If f is (approximately) equal to a polynomial of degree less than $J/2$ over the support of a k-level DaubJ wavelet W_m^k, then the k-level fluctuation value $\mathbf{f} \cdot \mathbf{W}_m^k$ is (approximately) zero.*

As with Property II above, this property implies that the Daub$J$ transform will produce a large number of small fluctuation values for a signal that is sampled from a smooth, many times continuously differentiable, signal. To put it another way, we can more closely approximate (obtain a better fit to) a wider range of signals if we can use higher degree polynomials with degree less than $J/2$, and yet still expect that the Daub$J$ transform will produce large numbers of small fluctuation values. As we showed in the previous chapter, when a wavelet transform produces a large number of small fluctuation values then we can obtain very effective compression and good noise removal.

One advantage of using a Daub$J$ wavelet with a larger value for J, say $J = 20$, is that there is an improvement in the resulting MRA for smoother signals (signals sampled from analog signals having more differentiability). For example, in Figure 2.6 we show the Daub20 MRA for the same signal analyzed previously with Haar and Daub4 wavelets. Notice that the Daub20 MRA is superior to both of these previous multiresolution analyses, especially for the lower resolution averaged signals.

We do not mean to suggest, however, that Daub20 wavelets are always the best. For example, for Signal 1 shown in Figure 1.4(a), the Haar wavelets do the best job of compression and noise removal (for reasons discussed in the previous chapter). As a simple comparison of the Haar, Daub4, and Daub20 wavelets, in Table 2.1 we list the minimum number of transform values needed to capture 99.99% of the energy in Signal 1. This table shows that the Haar transform does the best job, that the Daub4 transform is worse by a factor of 2, and that the Daub20 transform is the worst of all.

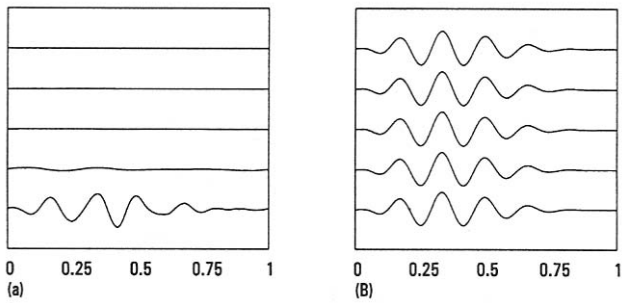

FIGURE 2.6
Daub20 MRA of the signal shown in Figure 1.1(a). The graphs are of the 10 averaged signals $\mathbf{A}^{10}$ through $\mathbf{A}^{1}$. Beginning with $\mathbf{A}^{10}$ on the top left down to $\mathbf{A}^{6}$ on the bottom left, then $\mathbf{A}^{5}$ on the top right down to $\mathbf{A}^{1}$ on the bottom right. Compare with Figures 1.3 and 2.3.

Table 2.1 Comparison of wavelet transforms of Signal 1

Wavelet transform	*Values for* 99.99% *energy*
Haar	51
Daub4	103
Daub20	205

The problem with Daub$J$ wavelets in terms of this signal is that these wavelets have longer supports than the Haar wavelets, all of their supports being at least twice as long, or longer, than the Haar wavelets. The Daub20 wavelets have the longest supports, with 1-level wavelets having supports of 20 time-units, and 2-level wavelets having supports of 58 time-units, and so on. Consequently, the percentage of Daub20 fluctuation values of this signal with significant energy will be high, due to the large number of Daub20 wavelets whose supports contain a point where a big jump in the signal's values occurs. A big jump in the signal's values induces corresponding jumps in the values of the scalar products that define the fluctuations, thus producing fluctuation values with significant energy.

Coiflets

We now turn to the description of another class of wavelets, the Coif$I$ wavelets. These wavelets are designed for the purpose of maintaining a close match between the trend values and the original signal values. Following a suggestion of Coifman, these wavelets were first constructed by Daubechies,

who called them "coiflets." All of the Coif$I$ wavelets are defined in a similar way; so we shall concentrate on the simplest case of Coif6 wavelets. The scaling numbers for the Coif6 scaling signals are listed in Table 2.2.

Table 2.2 Coif6 scaling numbers

$\alpha_1 = \frac{1-\sqrt{7}}{16\sqrt{2}}$,	$\alpha_2 = \frac{5+\sqrt{7}}{16\sqrt{2}}$,	$\alpha_3 = \frac{14+2\sqrt{7}}{16\sqrt{2}}$,
$\alpha_4 = \frac{14-2\sqrt{7}}{16\sqrt{2}}$,	$\alpha_5 = \frac{1-\sqrt{7}}{16\sqrt{2}}$,	$\alpha_6 = \frac{-3+\sqrt{7}}{16\sqrt{2}}$.

Using these scaling numbers, the first-level Coif6 scaling signals are defined by

$$\begin{aligned}
\mathbf{V}_1^1 &= (\alpha_3, \alpha_4, \alpha_5, \alpha_6, 0, 0, \ldots, 0, \alpha_1, \alpha_2) \\
\mathbf{V}_2^1 &= (\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6, 0, 0, \ldots, 0) \\
\mathbf{V}_3^1 &= (0, 0, \alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6, 0, 0, \ldots, 0) \\
&\vdots \\
\mathbf{V}_{N/2}^1 &= (\alpha_5, \alpha_6, 0, 0, \ldots, 0, \alpha_1, \alpha_2, \alpha_3, \alpha_4)
\end{aligned} \tag{2.28}$$

Notice that there are wrap-arounds for $\mathbf{V}_1^1$ and $\mathbf{V}_{N/2}^1$.

The Coif6 wavelet numbers are defined by

$$\beta_1 = \alpha_6,\ \beta_2 = -\alpha_5,\ \beta_3 = \alpha_4,\ \beta_4 = -\alpha_3,\ \beta_5 = \alpha_2,\ \beta_1 = -\alpha_1 \tag{2.29}$$

and these wavelet numbers determine the first-level Coif6 wavelets as follows:

$$\begin{aligned}
\mathbf{W}_1^1 &= (\beta_3, \beta_4, \beta_5, \beta_6, 0, 0, \ldots, 0, \beta_1, \beta_2) \\
\mathbf{W}_2^1 &= (\beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6, 0, 0, \ldots, 0) \\
\mathbf{W}_3^1 &= (0, 0, \beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6, 0, 0, \ldots, 0) \\
&\vdots \\
\mathbf{W}_{N/2}^1 &= (\beta_5, \beta_6, 0, 0, \ldots, 0, \beta_1, \beta_2, \beta_3, \beta_4)
\end{aligned} \tag{2.30}$$

As with the Coif6 scaling signals, there are wrap-arounds for the first and last wavelets.

The Coif6 scaling numbers satisfy the following identity

$$\alpha_1^2 + \alpha_2^2 + \alpha_3^2 + \alpha_4^2 + \alpha_5^2 + \alpha_6^2 = 1 \tag{2.31}$$

which implies that each Coif6 scaling signal has energy 1. Because of (2.29), it follows that each Coif6 wavelet also has energy 1. Furthermore, the

wavelet numbers satisfy

$$\beta_1 + \beta_2 + \beta_3 + \beta_4 + \beta_5 + \beta_6 = 0, \tag{2.32a}$$

$$0\beta_1 + 1\beta_2 + 2\beta_3 + 3\beta_4 + 4\beta_5 + 5\beta_6 = 0. \tag{2.32b}$$

These equations show that a Coif6 wavelet is similar to a Daub4 wavelet in that it will produce a zero fluctuation value whenever a signal is linear over its support. The difference between a Coif6 wavelet and a Daub4 wavelet lies in the properties of the scaling numbers. The Coif6 scaling numbers satisfy

$$\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 + \alpha_5 + \alpha_6 = \sqrt{2}, \tag{2.33a}$$

$$-2\alpha_1 - 1\alpha_2 + 0\alpha_3 + 1\alpha_4 + 2\alpha_5 + 3\alpha_6 = 0, \tag{2.33b}$$

$$(-2)^2\alpha_1 + (-1)^2\alpha_2 + 0^2\alpha_3 + 1^2\alpha_4 + 2^2\alpha_5 + 3^2\alpha_6 = 0. \tag{2.33c}$$

Equation (2.33a) implies, as usual, that Coif6 trend values are averages of successive values of a signal $\mathbf{f}$ (with wrap-around when $\mathbf{f} \cdot \mathbf{V}_1^1$ and $\mathbf{f} \cdot \mathbf{V}_{N/2}^1$ are computed). The second two equations, however, are entirely new. No Daub$J$ scaling numbers satisfy any equations of this type. These three equations have an important consequence. *When a signal consists of sample values of an analog signal, then a Coif6 transform produces a much closer match between trend subsignals and the original signal values than can be obtained with any of the DaubJ transforms.* By a close match between trends and signal values, we mean that the following approximations hold to a high degree of accuracy:

$$\mathbf{a}_m^1 \approx \sqrt{2}g(t_{2m}), \qquad \mathbf{a}_m^2 \approx 2g(t_{4m}) \tag{2.34}$$

Similar approximations will hold for higher levels, but the accuracy generally decreases as the number of levels increases.

As an example of (2.34), consider the signal graphed in Figure 2.2(a). This signal is obtained from 2^{14} sample values of the function

$$g(x) = 20x^2(1-x)^4 \cos 12\pi x \tag{2.35}$$

over the interval $[0, 1)$. When a 2-level Daub4 transform is performed on this signal then we obtain the graph shown in Figure 2.2(b). A 2-level Coif6 transform looks much the same. Computing the maximum error between the 2-level Daub4 trend values and samples of $2g(4x)$ over the interval $[0, .25)$, we obtain 3.76×10^{-3}. The maximum error in the Coif6 case is 4.84×10^{-7}, which is much smaller. For the 1-level transforms we find that the maximum error between the first trend and samples of $\sqrt{2}g(2x)$ is 8.87×10^{-4} in the Daub4 case, and 8.59×10^{-8} in the Coif6 case. This property of trends providing close approximations of the analog signal, which is shared by all the Coif$I$ transforms, provides a useful means of interpreting the trend subsignals.

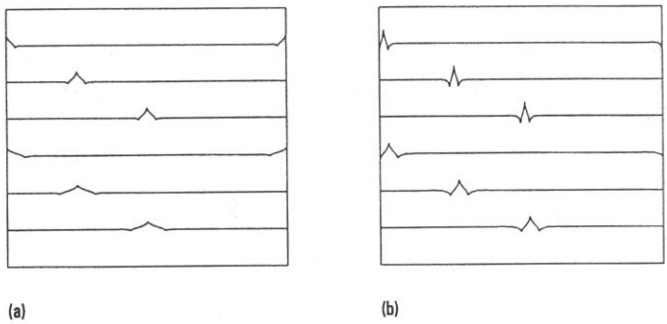

FIGURE 2.7
(a) The top 3 signals are 5-level Coif6 scaling signals, $\mathbf{V}_1^5$, $\mathbf{V}_8^5$, and $\mathbf{V}_{16}^5$. The bottom 3 signals are 6-level scaling signals $\mathbf{V}_1^6$, $\mathbf{V}_4^6$, and $\mathbf{V}_8^6$. (b) The top 3 signals are 5-level Coif6 wavelets, $\mathbf{W}_1^5$, $\mathbf{W}_8^5$, and $\mathbf{W}_{16}^5$. The bottom 3 signals are 6-level wavelets $\mathbf{W}_1^6$, $\mathbf{W}_4^6$, and $\mathbf{W}_8^6$.

Another interesting feature of Coif*I* scaling signals and wavelets is that their graphs are nearly symmetric. For example, in Figure 2.7 we graph several Coif6 scaling signals and wavelets. Notice how these Coif6 signals (especially if we discount the wrap-around effects) are much closer to being symmetric than the Daub4 signals graphed in Figure 2.1.

2.4 Compression of audio signals

One of the fundamental applications of wavelet transforms is the compression of signals. We outlined a basic method for wavelet transform compression in Section 1.5. In that section we focused on the Haar wavelet transform; in this section we shall work with the Daubechies wavelet transforms. We shall also discuss the problem of *quantization,* which we omitted from our first treatment of compression.

Recall that the basic method of wavelet transform compression consisted of setting equal to zero all transform values whose magnitudes lie below a threshold value. The compressed version of the signal consists of the significant, non-zero, values of the transform which survived the thresholding, along with a significance map indicating their indices. Decompression consists in using the significance map and the significant transform values to reconstruct the thresholded transform, and then performing an inverse wavelet transform to produce an approximation of the original signal. Compression works well when very few, high-energy, transform values capture most of the energy of the signal. For instance, consider again the two Sig-

nals 1 and 2 examined in Section 1.5. We saw that Signal 1 can be very effectively compressed using a Haar transform. This is revealed by the Energy map for its Haar transform in Figure 1.4(c), which shows that the Haar transform effectively captures most of the energy of Signal 1 in relatively few values.

None of the Daubechies transforms can do a better job compressing Signal 1. We have already examined why the Haar transform performs so well on Signal 1. In fact, the Haar transform and a related transform called the Walsh transform[3] have been used for many years as tools for compressing *piecewise constant* signals like Signal 1. We also saw, however, that Signal 2 does not compress particularly well using the Haar transform. This is explained by an examination of its Energy map shown in Figure 1.5(c). Let's instead try compressing Signal 2 using one of the Daubechies transforms. Signal 2 consists of $4096 = 2^{12}$ points; so we will use a 12-level transform, say a Coif30 transform. In Figure 2.8, we show the results of applying a Coif30 wavelet transform compression on Signal 2. It is interesting to compare this figure with Figure 1.4. It is clear that the 12-level Coif30 transform compresses Signal 2 just as well as the Haar transform compresses Signal 1. In fact, by using only the top 125 highest magnitude Coif30 transform values—which can be done by choosing a threshold of .00425—the compressed signal captures 99.99% of the energy of Signal 2. This compressed signal is shown in Figure 2.8(d). Since $4096/125 \approx 32$, the compressed signal achieves a 32:1 compression ratio. Here we are ignoring issues such as quantization and compression of the significance map. We now turn to a brief discussion of these deeper issues of compression; this initial treatment will be expanded upon in the next section.

Quantizing signal values, compressing the significance map

A digital audio signal typically consists of integer values that specify *volume levels*. The two most common ranges of volume levels are either $256 = 2^8$ volume levels, which require 8 bits to describe, or $65536 = 2^{16}$ volume levels, which require 16 bits to describe. An analog audio signal is *quantized* by a mapping from the recorded volume level to one of these two ranges of volume levels. Therefore, a discrete audio signal of length N will be initially described by either $8N$ or $16N$ bits, depending on which range of volume levels is used for recording. The 8-bit range is frequently used for voices in telephone transmission, where high fidelity is sacrificed for speed of transmission in order to accommodate the large number of signals that must be transmitted. The 16-bit range is frequently used for music, where high fidelity is most valued.

[3]The Walsh transform is described in Section 4.1.

Table 2.3 Encoding 16 volume levels using 4 bits

Volume level	*Encoding*
-24	1111
-21	1110
$\vdots$	$\vdots$
-1	1000
0	0000
1	0001
$\vdots$	$\vdots$
18	0110
21	0111

The most commonly employed quantization method for sampled analog signals is *uniform scalar quantization.* This method simply divides the range of volume levels into a fixed number of uniform width subintervals and rounds each volume level into the midpoint of the subinterval in which it lies. For instance, in Figure 2.9(a), we show a simple uniform scalar quantization map that encodes volume levels using 4 bits. The volume interval $[-24, 21]$ is divided into $16 = 2^4$ equal width subintervals, with all volumes that are below -24 being truncated to -24 and all volumes that are greater than 21 being truncated to 21. These 16 volume levels can be encoded as shown in Table 2.3. The asymmetry in this uniform quantization decreases as more subintervals, i.e., more bits, are used.

In order to take into account the number of bits used per point (bpp), which is either 8 bpp or 16 bpp in a quantized audio signal, we must also quantize the transform coefficients. That is, we must use only a finite number of bits to describe each transform value, and the number of bpp must be significantly less for the compressed signal than for the original signal.

As an initial example of handling quantization, consider again Signal 2 shown in Figure 2.8(a). This signal was generated from 4096 uniform samples of an analog signal. If this signal is uniformly scalar quantized with 16 bpp and played as an audio signal at a rate of 8820 samples per second,[4] multiplying its volume by a factor of 32000, then the resulting sound resembles two low notes played on a clarinet. The 16-bit quantized version of the signal has a graph that is almost identical to the one shown in Figure 2.8(a). If a Coif30 transform is performed on this quantized signal,

[4]Volume levels are sent to the sound system at a rate of 8820 values per second.

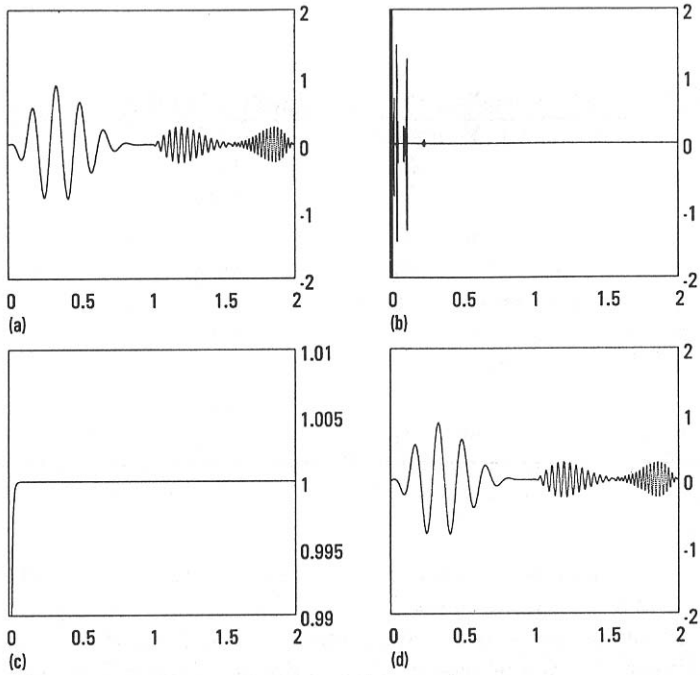

FIGURE 2.8
(a) Signal 2, 4096 values. (b) 12-level Coif30 transform. (c) Energy map of the Coif30 transform. (d) 32:1 compression of Signal 2, 99.99% of energy of Signal 2.

then a transform is produced which is virtually indistinguishable from the graph shown in Figure 2.8(b). To quantize this transform, we proceed as follows. The quantization map used is similar to the one shown in Figure 2.9(b). This is called *uniform quantization with a dead-zone.* The values in the subinterval $(-T, T)$ are the insignificant values whose magnitudes lie below a threshold value of T. Since these values will not be transmitted they are not encoded by the quantization. The remainder of the range of transform values lies in the two intervals $[-M, -T]$ and $[T, M]$, where M is the maximum for all the magnitudes of the transform values. These two intervals are divided into uniform width subintervals and each transform value is rounded into the midpoint of the subinterval containing it. For Signal 2, the value of M is 9.4, and a threshold value of $T = 9.4/2^7$ results in only 100 significant values. These significant values can then be encoded using 8 bits, 7 bits for the levels of magnitude and 1 bit for signs. As can be seen from Figure 2.8(b), the significant values of the transform lie in the interval $[0, .25)$. In fact, the significant values of the quantized transform lie among the first 256 values. Consequently the bits of value 1

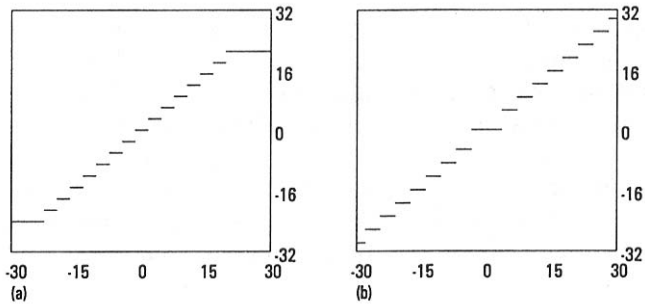

FIGURE 2.9
(a) Uniform, scalar quantization. (b) Uniform quantization, with a dead-zone containing the 0-level. The quantized values outside the dead-zone can be encoded using 4 bits.

in the significance map lie only within the first 256 bits. Therefore, this significance map can be transmitted in a very compressed form by just transmitting the first 256 bits, letting the decompression program supply the remaining bits, which are all 0. The result is that

$$\frac{256 + 8 \cdot 100}{4096} \approx 0.27 \text{ bpp}$$

are needed to transmit the compressed signal. This represents a compression ratio of 60:1. Even more important, when the decompressed signal is played, the resulting sound is indistinguishable from the original signal's sound. This is known as *perceptually lossless* compression.

This example was meant to illustrate the basic idea of quantization and its relation to wavelet transform compression of audio signals. In the next section we shall delve further into some of the fascinating complexities of quantization and compression of signals.

2.5 Quantization, entropy, and compression

In this section we shall examine some of the deeper questions involved with quantization and compression. One of the interesting features of quantizing wavelet transforms is the connection between structural properties of these transforms and basic ideas from information theory, such as entropy. This connection can be exploited to improve upon the basic compression results described in the preceding section.

Let's begin with a specific signal to be compressed, a recording of the

author speaking the word *greasy.* A graph of the intensity values of this recording is shown in Figure 2.10(a). Notice that this graph consists of three main sections, summarized in Table 2.4. This signal exhibits within a short time-span a number of different linguistic effects, such as a quick transition from a low pitch *gr* sound to a higher pitch *e* sound, and then another quick transition to a very chaotically oscillating *s* sound, with a final transition back to a high pitch *y* sound. These effects make *greasy* a challenging test signal for any compression method.

Table 2.4 Main sections of *greasy*

Section	*Time Interval*	*Sound*
1	$[0.00, 0.20]$	grea
2	$[0.25, 0.35]$	s
3	$[0.40, 0.60]$	y

The intensity values for this recording of *greasy* were quantized—using an 8-bit scalar quantization—as a part of the recording process. The sequences of bits encoding these 256 intensity values look like 01000001 or 11001000, and so on. A first bit of 1 indicates a negative intensity, while a first bit of 0 indicates a positive intensity. These bit sequences correspond to the integers $k = 0$ to $k = 255$, which we shall refer to as the *intensity levels.* To some extent this use of equal length bit sequences for all intensity levels is wasteful. This is indicated by the histogram of frequencies of occurrence of each intensity level shown in Figure 2.10(b). Since the most commonly occurring intensity level is the zero level, we can save bits if we encode this level with a single bit, such as 0.

By using shorter length bit sequences for the most commonly occurring intensity levels, and longer sequences for less commonly occurring intensity levels, we can reduce the total number of bits used. The idea is similar to Morse code where, for instance, the commonly occurring English letters *a* and *e* are encoded by the short sequences of dots and dashes $\cdot -$ and $\cdot$, respectively, while the less commonly occurring English letters *q* and *v* are encoded by the longer sequences $- - \cdot -$ and $\cdot \cdot \cdot -$, respectively. This procedure is made mathematically precise by fundamental results from the field known as *information theory.*

It is beyond the scope of this primer to provide a rigorous treatment of information theory. We shall just outline the basic ideas, and show how they apply to compressing *greasy.* Suppose that $\{p_k\}$ are the relative frequencies of occurrence of the intensity levels $k = 0$ through $k = 255$; that is, each p_k is an ordinate value in the histogram in Figure 2.10(b). Thus $p_k \geq 0$ for each k and $p_0 + p_1 + \cdots + p_{255} = 1$. These facts make it tempting to interpret

each number p_k as a probability for the occurrence of k. Although these numbers p_k are not probabilities, nevertheless, a deterministic law governing the production of the intensity levels in *greasy* is *a priori* unknown to us. In fact, section 2 of *greasy,* as an isolated sound, is very similar to the random static background noise considered in the next section. There are deterministic models for producing sections 1 and 3, involving combinations of sinusoidal signals, but these are *a posteriori* models based on the recorded sound itself. In any case, let's see what consequences follow from treating the numbers p_k as probabilities for the occurrence of the intensity levels k. Let $\mathcal{L}_k$ be the length of each bit sequence that is used to encode the intensity level k. Then the *average length* $\overline{\mathcal{L}}$ of a lossless encoding of the k's is defined to be

$$\overline{\mathcal{L}} = p_0\mathcal{L}_0 + p_1\mathcal{L}_1 + \cdots + p_{255}\mathcal{L}_{255}\,. \tag{2.36}$$

The famous Shannon Coding Theorem tells us that $\overline{\mathcal{L}}$ satisfies the following inequality [if $p_k = 0$, then $p_k \log_2(1/p_k)$ is set equal to 0]:

$$\overline{\mathcal{L}} \;\geq\; p_0 \log_2 \frac{1}{p_0} + p_1 \log_2 \frac{1}{p_1} + \cdots + p_{255} \log_2 \frac{1}{p_{255}}\,. \tag{2.37}$$

The quantity on the right side of (2.37) is called the *entropy* for the probabilities p_k.

Inequality (2.37) says that the average length of any lossless encoding of the intensity levels k cannot be less than the entropy of the probabilities p_k in the histogram for these intensity levels. Or another, more accurate, way of putting things is that *a lossless encoding technique, such as Huffman coding or arithmetic coding, which is based on the relative frequencies p_k in the histogram for the k's, cannot achieve an average length less than the entropy.* For obvious reasons, these lossless coding techniques are called *entropy codings.*

For *greasy,* the entropy is found to be 5.43. Therefore, it is impossible to make an entropy coding of the intensity levels for *greasy* with any set of bit sequences whose average length is less than 5.43 bits. It should also be noted that, using either Huffman coding or arithmetic coding, it is possible to get within 1 bit or less of the entropy. For reasons of space, we shall not describe the methodologies of these encoding techniques; for another matter, these techniques are very well-known and there are many excellent descriptions of them to be found in the references for this chapter. Huffman codes are guaranteed, by the way in which they are constructed, to always achieve an average length that is within 1 bit of the entropy; while arithmetic codes can get asymptotically close to the entropy as the number of values to be encoded increases. Therefore, as a rough estimator of the average length of a well-chosen lossless encoding of the intensity levels we shall add 0.5 to the entropy. Using this estimator, we find that the 16,384

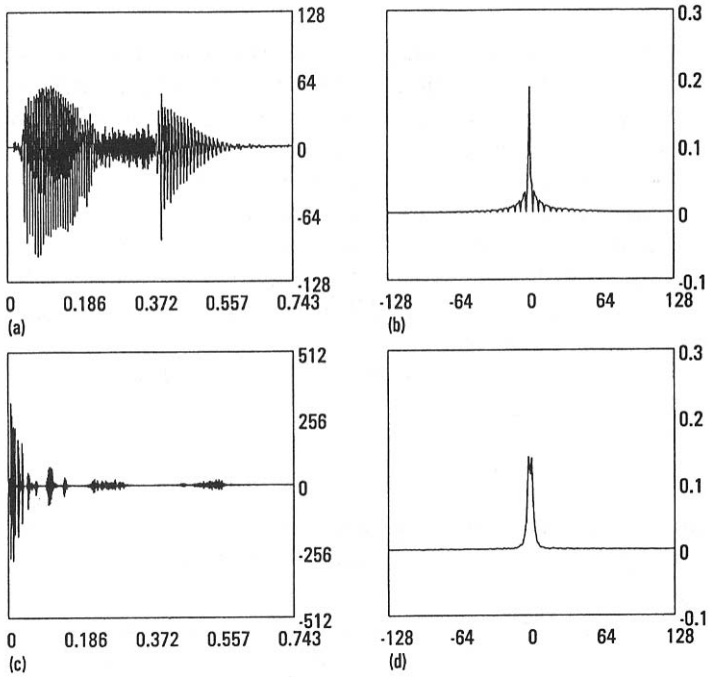

FIGURE 2.10
(a) The signal, *greasy*. (b) Histogram of 8-bit quantization of intensity levels of *greasy*. (c) 14-level Coif30 transform of *greasy*. (d) Histogram of 8-bit dead-zone quantization of the transform values.

points of *greasy* can be expected to be entropy encoded with a total of $16,384 \times 5.93$ bits, i.e., using about $97,000$ bits. This is not a particularly effective compression, since it still represents 5.93 bpp versus 8 bpp for the original signal.

The basis of wavelet transform encoding, as we explained in the last section, is to allow some inaccuracy resulting from quantizing transform values in order to achieve greater compression than by lossless methods. For instance, in Figure 2.10(c), we show a 14-level Coif30 transform of the *greasy* recording, and in Figure 2.10(d) we show a histogram of an 8-bit dead-zone quantization of this transform. Comparing the two histograms in Figure 2.10, we see that the histogram for the dead-zone quantization of the transform values is more narrowly concentrated than the histogram for the scalar quantization of the signal values. In fact, the entropy for the histogram in Figure 2.10(d) is 4.34, which is smaller than the entropy 5.43 for the histogram in Figure 2.10(c). Using our estimator for average coding length, we estimate that a lossless encoding of the quantized transform values will have an average length of 4.84. Consequently, the 3922 non-zero

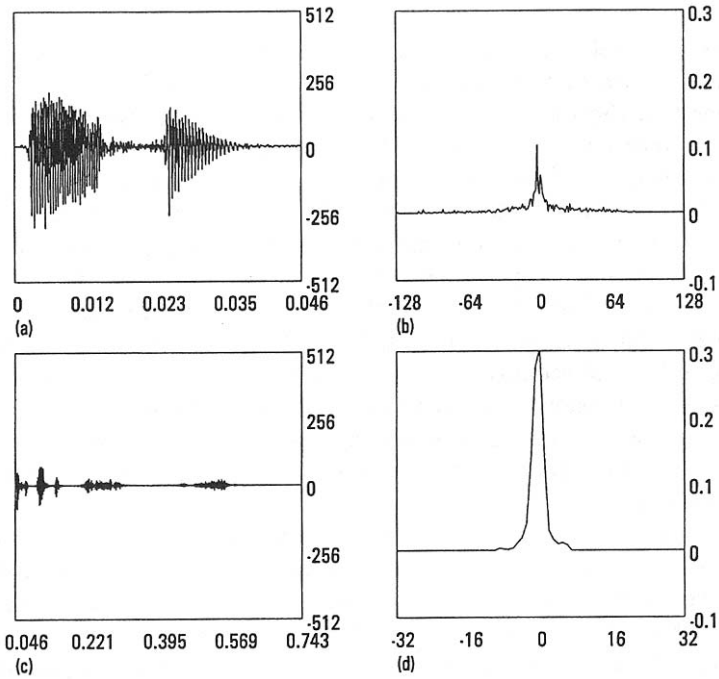

FIGURE 2.11
(a) Fourth trend of 4-level Coif30 transform of *greasy*. (b) Histogram of 8-bit dead-zone quantization of fourth trend. (c) Fluctuations of 4-level Coif30 transform. (d) Histogram of 6-bit dead-zone quantization of fluctuations.

quantized transform values can be expected to be losslessly encoded with about $3922 \times 4.84 \approx 18,922$ bits. This is a significant reduction in the number of bits; even if absolutely no compression was done on the 16,384 bits in the significance map, there would still be a large improvement in compression over a lossless compression of *greasy*. It is very important to note that—even though there is some error introduced by the quantization of the transform values—when an inverse transform is performed, the resulting signal is extremely difficult to distinguish from the original when played over a computer sound system.

As another example of the possibilities available with wavelet transform compression, we shall examine a slightly different approach to compressing *greasy*. In Figures 2.11(a) and 2.11(c), we show a 4-level Coif30 transform of *greasy*. The fourth trend is graphed in Figure 2.11(a) and the 4 fluctuations are graphed in Figure 2.11(c). An 8-bit dead-zone quantization of the fourth trend has the histogram shown in Figure 2.11(b), with an entropy of 6.19. There are 793 non-zero coefficients; so encoding requires about

$793 \times 6.69 \approx 5305$ bits. To quantize the fluctuation values we chose to use only 6 bits, not 8 bits as we used for the 14-level transform. A comparison of Figures 2.10(c) and 2.11(c) makes it apparent why we did this. More intensity levels are needed with the 14-level transform in order to ensure accuracy because of the many larger fluctuation values at higher levels [due to the higher peaks on the left side of Figure 2.10(c)]. This 6-bit dead-zone quantization of the four fluctuations has the histogram shown in Figure 2.11(d), with an entropy of 2.68. Since there are 2892 non-zero quantized values, an encoding requires about $2892 \times 3.18 \approx 9197$ bits. The total number of bits estimated for this 4-level transform compression is about $14,502$ bits. This compares favorably with the $18,982$ bits estimated for the 14-level transform compression. The improvement in compression for the 4-level transform is due to the decrease in entropy from 4.34 to 2.68 for the first four fluctuations brought about by the change made in the quantization.

This last example only begins to suggest some of the many possibilities available for adaptation of the basic wavelet transform compression procedure. For instance, one possibility is to compute an entropy for a separate quantization of each fluctuation and separately encode these quantized fluctuations. This does generally produce improvements in compression. For example, suppose a 4-level Coif30 transform of *greasy* is quantized using 8 bpp for the trend and 6 bpp for the four fluctuations, and separate entropies are calculated for the trend and for each of the four fluctuations. Then the estimated total number of bits needed is $11,305$. This is an improvement over the $14,502$ bits previously estimated.

2.6 Denoising audio signals

As we saw in Section 1.6, the problems of compression and noise removal are closely related. If a wavelet transform can effectively capture the energy of a signal in a few high-energy transform values, then additive noise can be effectively removed as well. We introduced a basic method, called *thresholding,* for removing noise, and illustrated this method using the Haar transform. Now, in this section, we shall threshold the Daubechies wavelet transforms. Besides discussing a simple illustrative example, we also shall give some justification for why thresholding works well for the random noise often encountered in signal transmission, and provide an example of denoising when the noise is a combination of pop noise and random noise.

Let's quickly review the basic steps for removing additive noise using the Threshold Method. A threshold value T is chosen for which all transform values that are lesser in magnitude than T are set equal to zero. By

performing an inverse wavelet transform on the thresholded transform, an estimation of the original uncontaminated signal is obtained. In Chapter 1 we saw that the Haar transform was able to effectively remove the noise from Signal A, as shown in Figure 1.6. Signal A was created by adding random noise to Signal 1, whose graph is shown in Figure 1.4. Because of the connection between compression and noise removal, and because the Haar transform is the most effective transform for compressing Signal 1, it follows that the Haar transform is also the most effective transform for denoising Signal A.

With the other noisy signal examined in Section 1.6, Signal B, we found that the Haar transform did a rather poor job of denoising. The explanation for this lies again in the connection between compression and threshold denoising. Signal B was created by adding random noise to Signal 2, but the Haar transform is not an effective tool for compressing Signal 2; it produces too many low magnitude transform values which are obscured by the noise. We saw, however, in Section 2.4 that the Coif30 transform is a very effective tool for compressing Signal 2. Therefore, let's apply it to denoising Signal B.

In Figure 2.12 we show the basic steps in a threshold denoising of Signal B using a 12-level Coif30 transform. Comparing the 12-level Coif30 transforms of Signal B and Signal 2, we see that the addition of the noise has contributed a large number of small magnitude, low-energy values to the transform of Signal 2. Nevertheless, most of the high-energy values of the transform of Signal 2 are still plainly visible in the transform of Signal B, although their values have been altered slightly by the added noise. Therefore, we can eliminate the noise using thresholding, as indicated by the two horizontal lines in Figure 2.12(b). All transform values between ± 0.2 are set equal to zero, and this produces the thresholded transform shown in Figure 2.12(c). Comparing this thresholded transform with the transform of Signal 2, shown in Figure 2.8(b), we can see that thresholding has produced a fairly close match. The most noticeable difference is a loss of a small number of values located near 0.25, which had such small magnitudes that they were obscured by the addition of the noise. Since the two transforms are such a reasonably close match, it follows that the inverse transform of the thresholded transform produces a denoised signal that is a close match of Signal 2. In fact, the RMS Error between the denoised signal and Signal 2 is 0.14, which is a four-fold decrease in the RMS Error of 0.57 between Signal B and Signal 2.

Choosing a threshold value

One of the most attractive features of wavelet threshold denoising is that, for the type of random noise frequently encountered in signal transmission,

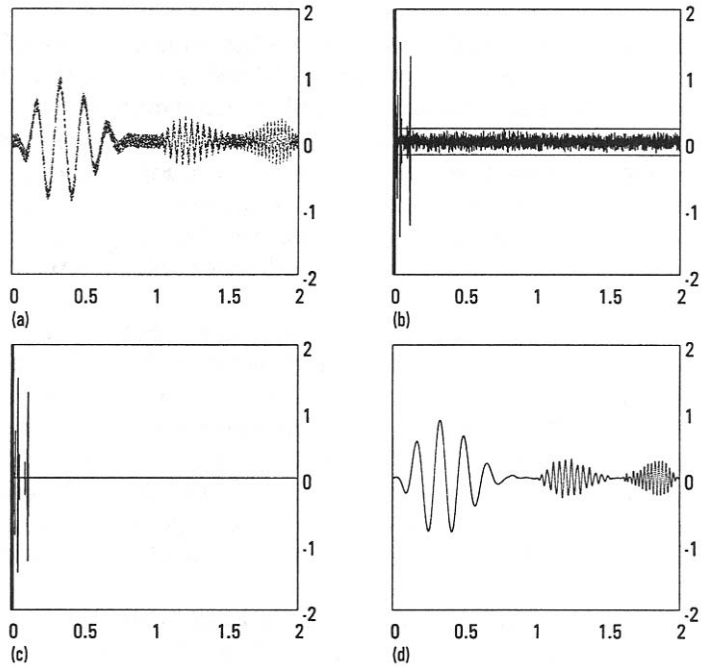

FIGURE 2.12
(a) Signal B. (b) 12-level Coif30 transform, with thresholding indicated by two horizontal lines at ± 0.2. (c) Thresholded transform. (d) Denoised signal; compare with Figures 2.8(a) and 1.6(d).

it is possible to automatically choose a threshold for denoising *without any prior knowledge of the signal.* In Figure 2.13(a) we show an example of this type of random noise; it is called *Gaussian noise.* A technical definition of Gaussian random noise is beyond the scope of this book; in this subsection we shall only give an informal introduction to some of the main ideas.

If the noise shown in Figure 2.13(a) is played over a sound generator, it produces the familiar static sound that can be heard in noisy radio transmissions and analog recordings. A histogram of frequencies of occurrence of intensity levels for this noise is shown in Figure 2.13(b). The bell-shaped curve that this histogram approximates is an indication that this noise is Gaussian. Using elementary statistical formulas on the noise values, we can estimate the mean μ and standard deviation σ of the probability density function (the bell-shaped curve) that this histogram approximates. We find that the mean is approximately 0 and that the standard deviation is approximately 0.579. One of the consequences of the Conservation of Energy Property of the Daubechies wavelet transforms—more precisely, the

Table 2.5 Comparison of three noise histograms

Threshold	*% below, Theory*	*% below, 2.13(a)*	*% below, 2.13(c)*
σ	68.72	68.87	69.94
2σ	95.45	95.70	95.39
3σ	99.73	99.74	99.66
4σ	99.99	100.00	99.98

orthogonality of the matrix form for these transforms[5]—is that they preserve the Gaussian nature of the noise. For example, in Figure 2.13(c) we show a Coif30 wavelet transform of the random noise in Figure 2.13(a). Its histogram in Figure 2.13(d) approximates a bell-shaped curve, and the mean and standard deviation of the transformed noise are calculated to be approximately 0 and 0.579, the same values that we found for the original noise.

These facts imply that the transformed noise values will be similar in magnitude to the original noise values and, what is more important, that *a large percentage of these values will be smaller in magnitude than a threshold equal to a large enough multiple of the standard deviation* σ. To see this last point more clearly, consider the data shown in Table 2.5. In this table, the first column contains four thresholds which are multiples of the standard deviation $\sigma = 0.579$. The second column lists the percentages of values of random numbers having magnitudes that are less than these thresholds, assuming that these random numbers obey a Gaussian normal probability law with mean 0 and standard deviation σ.[6] The third and fourth columns contain the percentages of the magnitudes from the noise signals in Figure 2.13(a) and 2.13(c) which lie below the thresholds.

Based on the results of this table, we shall use the following formula:

$$T = 4.5\sigma \tag{2.38}$$

for setting the threshold value T. In our next example, we shall make use of (2.38) for choosing a threshold. The standard deviation σ can be estimated from a portion of the transform which consists largely of noise values. Generally this is the case with the first level fluctuation because the first level fluctuation values from the original signal are typically very small. When Formula (2.38) is used we expect that well over 99% of the

[5]Orthogonality of the Daubechies wavelet transforms was discussed in Section 2.2.

[6]That is, the probability of a number x having magnitude less than T is equal to the area under the curve $\frac{1}{\sigma\sqrt{2\pi}}e^{-x^2/2\sigma^2}$ from $-T$ to T.

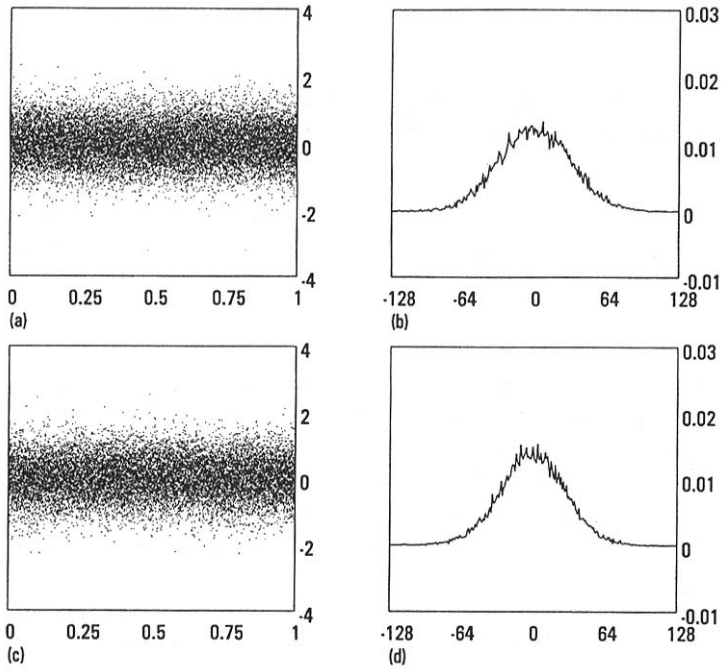

FIGURE 2.13
(a) Gaussian noise. (b) Histogram of the noise. (c) Coif30 transform of the noise. (d) Histogram of the transformed noise.

noise values will be removed from the transform, and this requires essentially no knowledge of the original, uncontaminated signal. Preventing this thresholding from removing too many transform values from the original signal, however, depends on how well the transform compresses the energy of the signal into a few high-magnitude values which stand out from the threshold. With the Daubechies transforms this will occur with signals that are sampled from analog signals that are smooth. This is because so many of the fluctuation values are small, and that leaves only a few high-magnitude transform values to account for the energy. Of course it is always possible—by adding noise with a sufficiently high standard deviation—to produce such a high threshold that the signal's transform values are completely wiped out by thresholding. Wavelet thresholding is powerful, but it cannot perform magic.

Removing pop noise and background static

We close this section by discussing another example of noise removal. In Figure 2.14(a) we show the graph of a noisy signal. The signal consists of

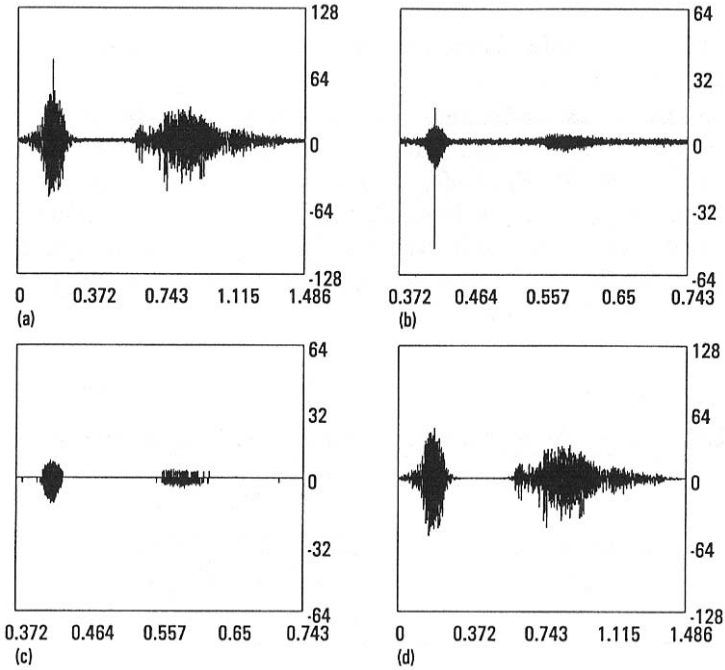

FIGURE 2.14
(a) Noisy whistle. (b) Second fluctuation of the 15-level Coif18 transform. (c) Second fluctuation of the denoised transform. (d) Denoised whistle.

Gaussian random noise and pop noise added to a recording of the author making a whistling sound. The pop noise consists of the spike located near 0.186, which is plainly visible in the graph. When this signal is played over a computer sound system, the whistle can be plainly heard but there is an annoying static background and a rather loud, short, popping sound near the beginning of the whistle. Using wavelet techniques we can remove essentially all of the random noise and greatly reduce the volume of the pop noise.

To remove the noise, we use a Coif18 wavelet transform of the signal. In Figure 2.14(b) we show the second fluctuation subsignal of a Coif18 transform of the noisy signal. The other fluctuation subsignals are similar in appearance; so by describing how to modify this particular fluctuation we shall be describing how to handle the other fluctuations as well. The random noise, or background static, appears as random oscillations in the wavelet transform as we noted above. In fact, on the interval $[.46, .51]$ the fluctuation values appear purely random; so we make an estimate of the standard deviation of the transform noise over this subinterval, obtaining

$\sigma \approx 0.617$. By Formula (2.38), we should set T equal to 2.7765. Rounding up slightly, we set $T = 2.78$.

In contrast to the random noise, the pop noise produces unusually large magnitude fluctuation values that stand out from the vast majority of the fluctuation values. We shall refer to these unusually large fluctuation values as *outliers*. For example, in Figure 2.14(b) there are two outliers that are clearly visible in the second fluctuation. They appear as two spikes on the left side of the figure. To remove the pop noise we must eliminate these outliers from the transform.

Table 2.6 Acceptance bands for fluctuations of noisy signal

Fluctuation	*Acceptance band*
1	$(-\infty, -T] \cup [T, \infty)$
2	$(-12.5, -T] \cup [T, 9)$
3	$(-\infty, -T] \cup [T, \infty)$
4	$(-67, -T] \cup [T, 67)$
5 to 15	$(-\infty, -T] \cup [T, \infty)$

To remove both the random noise values and the outliers from the pop noise, we set *acceptance bands* for each fluctuation subsignal. Values that lie in the acceptance band for each fluctuation are retained and all other values are set equal to zero. These acceptance bands, which were obtained by a visual inspection of the transform, are summarized in Table 2.6. The second fluctuation of the modified, denoised transform is shown in Figure 2.14(c). By performing an inverse transform on the denoised transform, we produce the denoised whistle signal shown in Figure 2.14(d). When this denoised signal is played on a computer's sound system, the background static is completely gone and the volume of the pop noise, although still audible, is greatly diminished.

No claim is being made here that the method used for removing the pop noise is in any way optimal. There are two problems with the approach used. First, we only removed outliers from the fluctuation levels 2 and 4, and ignored higher levels. An examination of the transform reveals outliers for several other levels, such as 5 and 6. A more effective denoising would remove these spikes as well. Furthermore, actually removing the spikes completely may not be the best approach; reducing their size based on an estimate of nearby fluctuation values might work better. In any case, this example was described in order to indicate some of the flexibility available in wavelet denoising. More detailed discussions of denoising methods can be found in the references.

2.7 Two-dimensional wavelet transforms

Up till now we have been working with one-dimensional signals, but wavelet analysis can be done in any number of dimensions. The essential ideas, however, are revealed in two dimensions. In this section we shall begin our treatment of 2D wavelet analysis. Many of the basic ideas are similar to the 1D case; so we shall not repeat similar formulas but rather focus on the new ideas that are needed for the 2D case. In subsequent sections we shall describe various applications of 2D wavelet analysis to compression of images, denoising of images and other types of image enhancements and image analysis.

Discrete images

The 2D data that we shall be working with are discrete images. A *discrete image* $\mathbf{f}$ is an array of M rows and N columns of real numbers:

$$\mathbf{f} = \begin{pmatrix} f_{1,M} & f_{2,M} & \cdots & f_{N,M} \\ \vdots & \vdots & \ddots & \vdots \\ f_{1,2} & f_{2,2} & \cdots & f_{N,2} \\ f_{1,1} & f_{2,1} & \cdots & f_{N,1} \end{pmatrix}. \tag{2.39}$$

The *values* of $\mathbf{f}$ are the MN real numbers $\{f_{j,k}\}$. It should be noted that the way in which the values of $\mathbf{f}$ are displayed in the array on the right side of (2.39) is not the most commonly used one. We chose to display the values of $\mathbf{f}$ in this way because it corresponds well with the case where $\mathbf{f}$ is an array of sample values:

$$\mathbf{f} = \begin{pmatrix} g(x_1, y_M) & g(x_2, y_M) & \cdots & g(x_N, y_M) \\ \vdots & \vdots & \ddots & \vdots \\ g(x_1, y_2) & g(x_2, y_2) & \cdots & g(x_N, y_2) \\ g(x_1, y_1) & g(x_2, y_1) & \cdots & g(x_N, y_1) \end{pmatrix} \tag{2.40}$$

of a function $g(x, y)$ at the sample points (x_j, y_k) in the Cartesian coordinate plane. Just as with discrete 1D signals, it is frequently the case that a discrete 2D image is obtained from samples of some function $g(x, y)$.

It is often helpful to view a discrete image in one of two other ways. First, as a single column consisting of M signals having length N,

$$\mathbf{f} = \begin{pmatrix} \mathbf{f}_M \\ \vdots \\ \mathbf{f}_2 \\ \mathbf{f}_1 \end{pmatrix} \tag{2.41}$$

with the rows being the signals

$$\begin{aligned}\mathbf{f}_M &= (f_{1,M}, f_{2,M}, \ldots, f_{N,M})\\ &\vdots\\ \mathbf{f}_2 &= (f_{1,2}, f_{2,2}, \ldots, f_{N,2})\\ \mathbf{f}_1 &= (f_{1,1}, f_{2,1}, \ldots, f_{N,1}).\end{aligned}$$

Second, as a single row consisting of N signals of length M, written as columns,

$$\mathbf{f} = (\mathbf{f}^1, \mathbf{f}^2, \ldots, \mathbf{f}^N) \tag{2.42}$$

with the columns being the signals

$$\mathbf{f}^1 = \begin{pmatrix} f_{1,M} \\ \vdots \\ f_{1,2} \\ f_{1,1} \end{pmatrix}, \mathbf{f}^2 = \begin{pmatrix} f_{2,M} \\ \vdots \\ f_{2,2} \\ f_{2,1} \end{pmatrix}, \ldots, \mathbf{f}^N = \begin{pmatrix} f_{N,M} \\ \vdots \\ f_{N,2} \\ f_{N,1} \end{pmatrix}.$$

Notice that, because of our somewhat peculiar notation, the row index for each column increases from bottom to top (rather than from top to bottom, which is more common notation in image processing).

As an example of the utility of Formulas (2.41) and (2.42), we consider the calculation of the energy of a discrete image. The *energy* $\mathcal{E}_{\mathbf{f}}$ of a discrete image $\mathbf{f}$ is defined to be the sum of the squares of all of its values. Because of (2.41) it follows that $\mathcal{E}_{\mathbf{f}}$ is the sum of the energies of all of the row signals:

$$\mathcal{E}_{\mathbf{f}} = \mathcal{E}_{\mathbf{f}_1} + \mathcal{E}_{\mathbf{f}_2} + \cdots + \mathcal{E}_{\mathbf{f}_M}.$$

Or, because of (2.42), it follows that $\mathcal{E}_{\mathbf{f}}$ is also the sum of the energies of all of the column signals:

$$\mathcal{E}_{\mathbf{f}} = \mathcal{E}_{\mathbf{f}^1} + \mathcal{E}_{\mathbf{f}^2} + \cdots + \mathcal{E}_{\mathbf{f}^N}.$$

One consequence of these last two identities is that the 2D wavelet transforms defined below have the property of conserving the energy of discrete images.

2D wavelet transforms

A 2D wavelet transform of a discrete image can be performed whenever the image has an even number of rows and an even number of columns. A 1-level wavelet transform of an image $\mathbf{f}$ is defined, using any of the 1D wavelet transforms that we have discussed, by performing the following two steps:

Step 1. Perform a 1-level, 1D wavelet transform, on each row of $\mathbf{f}$, thereby producing a new image.

Step 2. On the new image obtained from Step 1, perform the same 1D wavelet transform on each of its columns.

It is not difficult to show that Steps 1 and 2 could be done in reverse order and the result would be the same. A 1-level wavelet transform of an image $\mathbf{f}$ can be symbolized as follows:

$$\mathbf{f} \longmapsto \begin{pmatrix} \mathbf{h}^1 & | & \mathbf{d}^1 \\ — & & — \\ \mathbf{a}^1 & | & \mathbf{v}^1 \end{pmatrix} \tag{2.43}$$

where the subimages $\mathbf{h}^1$, $\mathbf{d}^1$, $\mathbf{a}^1$, and $\mathbf{v}^1$ each have $M/2$ rows and $N/2$ columns. We shall now discuss the nature of each of these subimages.

The subimage $\mathbf{a}^1$ is created by computing trends along rows of $\mathbf{f}$ followed by computing trends along columns; so it is an averaged, lower resolution version of the image $\mathbf{f}$. For example, in Figure 2.15(a) we show a simple test image of an octagon, and in Figure 2.15(b) we show its 1-level Coif6 transform. The $\mathbf{a}^1$ subimage appears in the lower left quadrant of the Coif6 transform, and it is clearly a lower resolution version of the original octagon image. Since a 1D trend computation is $\sqrt{2}$ times an average of successive values in a signal, and the 2D trend subimage $\mathbf{a}^1$ was computed from trends along both rows and columns, it follows that each value of $\mathbf{a}^1$ is equal to 2 times an average of a small square containing adjacent values from the image $\mathbf{f}$. A useful way of expressing the values of $\mathbf{a}^1$ is as scalar products of the image $\mathbf{f}$ with scaling signals, as we did in the 1D case; we shall say more about this later in the section.

The $\mathbf{h}^1$ subimage is created by computing trends along rows of the image $\mathbf{f}$ followed by computing fluctuations along columns. Consequently, wherever there are horizontal edges in an image, the fluctuations along columns are able to detect these edges. This tends to emphasize the horizontal edges, as can be seen clearly in Figure 2.15(b) where the subimage $\mathbf{h}^1$ appears in the upper left quadrant. Furthermore, notice that vertical edges, where the octagon image is constant over long stretches, are removed from the subimage $\mathbf{h}^1$. This discussion should make it clear why we shall refer to this subimage as the first *horizontal fluctuation.*

The subimage $\mathbf{v}^1$ is similar to $\mathbf{h}^1$, except that the roles of horizontal and vertical are reversed. In Figure 2.15(b) the subimage $\mathbf{v}^1$ is shown in the lower right quadrant. Notice that horizontal edges of the octagon are erased, while vertical edges are emphasized. This is typically the case with $\mathbf{v}^1$, which we shall refer to as the first *vertical fluctuation.*

Finally, there is the first *diagonal fluctuation,* $\mathbf{d}^1$. This subimage tends to emphasize diagonal features, because it is created from fluctuations along

both rows and columns. These fluctuations tend to erase horizontal and vertical edges where the image is relatively constant. For example, in Figure 2.15(b) the diagonal fluctuation appears in the upper right quadrant of the image, and it is clear that diagonal details are emphasized while horizontal and vertical edges are erased.

It should be noted that the basic principles discussed previously for 1D wavelet analysis still apply here in the 2D setting. For example, the fact that fluctuation values are generally much smaller than trend values is still true. In the wavelet transform shown in Figure 2.15(b), for instance, the fluctuation subimages $\mathbf{h}^1$, $\mathbf{v}^1$, and $\mathbf{d}^1$ have significantly smaller values than the values in the trend subimage $\mathbf{a}^1$. In fact, in order to make the values for $\mathbf{h}^1$, $\mathbf{v}^1$, and $\mathbf{d}^1$ visible, they are displayed on a logarithmic intensity scale, while the values for the trend subimage $\mathbf{a}^1$ are displayed using an ordinary, linear scale.

Furthermore, 2D wavelet transforms enjoy the Conservation of Energy property. As noted above, the energy of an image is the sum of the energies of each of its rows or each of its columns. Since the 1D wavelet transforms of the rows, performed in Step 1, preserve the row energies, the image obtained in Step 1 will have the same energy as the original image. Likewise, since the 1D wavelet transforms of the columns preserve their energies, it follows that the transform obtained in Step 2 has the same energy as the image from Step 1. Thus the 1-level wavelet transform has the same energy as the original image. For example, the energy of the octagon image in Figure 2.15(a) is 3919.0625, while the energy of its 1-level Coif6 transform is 3919.0622; the slight discrepancy between the two energies is attributable to the inevitable rounding error that arises from finite precision computer calculations.

As in 1D, multiple levels of 2D wavelet transforms are defined by repeating the 1-level transform of the previous trend. For example, a 2-level wavelet transform is performed by computing a 1-level transform of the trend subimage $\mathbf{a}^1$ as follows:

$$\mathbf{a}^1 \longmapsto \begin{pmatrix} \mathbf{h}^2 & | & \mathbf{d}^2 \\ — & & — \\ \mathbf{a}^2 & | & \mathbf{v}^2 \end{pmatrix}.$$

The 1-level fluctuations $\mathbf{h}^1$, $\mathbf{d}^1$, and $\mathbf{v}^1$ remain unchanged. In Figure 2.15(c) we show a 2-level Coif6 transform of the octagon image. In general, a k-level transform is defined by performing a 1-level transform on the previous trend $\mathbf{a}^{k-1}$. In Figure 2.15(d) we show a 3-level Coif6 transform of the octagon image.

It is interesting to compare the successive levels of the Coif6 transforms in Figure 2.15. Notice how it appears that we are systematically decomposing the original octagon image by peeling off edges; and these edges are retained

within the fluctuation subimages. This aspect of wavelet transforms plays a major role in the fields of *image recognition* and *image enhancement.* In Section 2.11 we shall discuss a few examples from these fields.

Besides Conservation of Energy these 2D wavelet transforms also perform a Compaction of Energy. For example, for the octagon image in Figure 2.15 most of the energy of the image is successively localized into smaller and smaller trend subimages, as summarized in Table 2.7. Notice, for example, that the third trend $\mathbf{a}^3$, which is 64 times smaller than $\mathbf{f}$ in terms of numbers of values, still contains over 96% of the total energy. In accordance with the Uncertainty Principle, however, some of the energy has leaked out into the fluctuation subimages. Consequently, in order to obtain an accurate approximation of $\mathbf{f}$, some of the highest energy fluctuation values—such as the ones that are visible in Figure 2.15(d)—would have to be included along with the third trend values when performing an inverse transform. This Compaction of Energy property, as in the 1D case, provides the foundation for the methods of compression and denoising that we shall discuss in subsequent sections.

Table 2.7 Compaction of energy of octagon image

Image	*Energy*	*% Total Energy*
$\mathbf{f}$	3919.06	100.00
$\mathbf{a}^1$	3830.69	97.75
$\mathbf{a}^2$	3811.06	97.22
$\mathbf{a}^3$	3777.55	96.39

2D wavelets and scaling images

As in the 1D case, the various levels of a wavelet transform can be computed via scalar products of the image $\mathbf{f}$ with elementary images called *scaling images* and *wavelets.* A scalar product of two images $\mathbf{f}$ and $\mathbf{g}$, both having M rows and N columns, is defined by

$$\mathbf{f} \cdot \mathbf{g} = f_{1,1}g_{1,1} + f_{1,2}g_{1,2} + \cdots + f_{N,M}g_{N,M}. \tag{2.44}$$

In other words, $\mathbf{f} \cdot \mathbf{g}$ is the sum of all the products of similarly indexed values of $\mathbf{f}$ and $\mathbf{g}$.

To see how this scalar product operation relates to wavelet transforms, let's consider the 1-level horizontal fluctuation $\mathbf{h}^1$. This subimage is defined by separate calculations of trends along rows and fluctuations along columns. It follows that the values of $\mathbf{h}^1$ are computed via scalar products with wavelets obtained by multiplying values of 1D scaling signals along

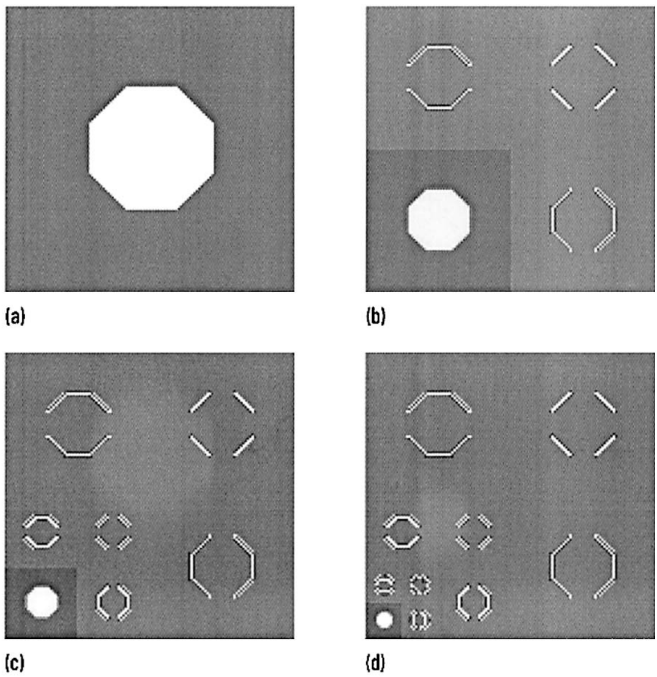

FIGURE 2.15
(a) Octagon image. (b) 1-level Coif6 transform. (c) 2-level Coif6 transform. (d) 3-level Coif6 transform.

rows by values of 1D wavelets along columns. Each such wavelet is denoted by $\mathbf{V}_m^1 \otimes \mathbf{W}_n^1$, which is called a *tensor product* of the 1D scaling signal $\mathbf{V}_m^1$ and 1D wavelet $\mathbf{W}_n^1$. For instance, if we are computing a 2D Haar wavelet transform, then $\mathbf{V}_1^1 \otimes \mathbf{W}_1^1$ is defined as follows:

$$\mathbf{V}_1^1 \otimes \mathbf{W}_1^1 = \begin{pmatrix} 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & 0 \\ -1/2 & -1/2 & 0 & \dots & 0 & 0 \\ 1/2 & 1/2 & 0 & \dots & 0 & 0 \end{pmatrix}.$$

Since each Haar scaling signal $\mathbf{V}_m^1$ is a translation by $2(m-1)$ time-units of $\mathbf{V}_1^1$, and each Haar wavelet $\mathbf{W}_n^1$ is a translation by $2(n-1)$ time-units of $\mathbf{W}_1^1$, it follows that $\mathbf{V}_m^1 \otimes \mathbf{W}_n^1$ is a translation by $2(m-1)$ units along the horizontal and $2(n-1)$ units along the vertical of $\mathbf{V}_1^1 \otimes \mathbf{W}_1^1$. Notice that the Haar wavelet $\mathbf{V}_1^1 \otimes \mathbf{W}_1^1$ has energy 1 and an average value of 0, as do all the other Haar wavelets $\mathbf{V}_m^1 \otimes \mathbf{W}_n^1$. Furthermore, the support of the Haar wavelet $\mathbf{V}_1^1 \otimes \mathbf{W}_1^1$ is a 2 by 2 square, and so the support of each

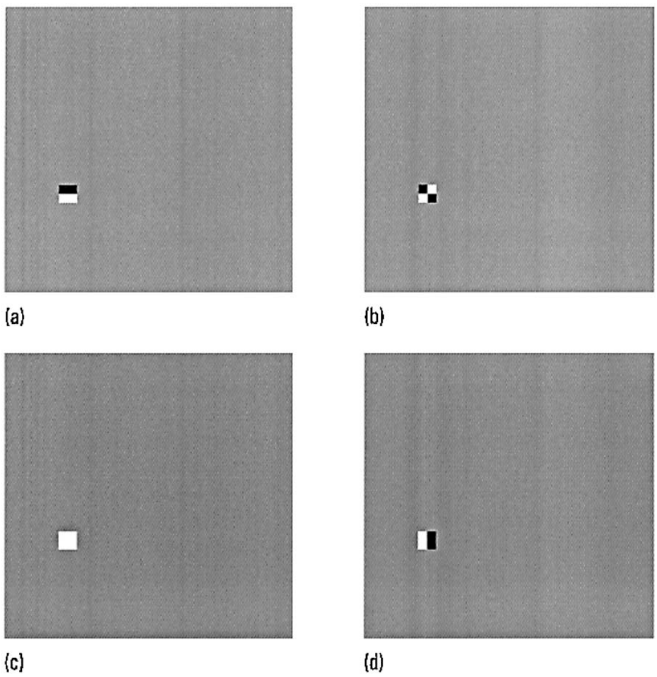

FIGURE 2.16
(a) Haar wavelet $\mathbf{V}_3^2 \otimes \mathbf{W}_5^2$. (b) Haar wavelet $\mathbf{W}_3^2 \otimes \mathbf{W}_5^2$. (c) Haar scaling image $\mathbf{V}_3^2 \otimes \mathbf{V}_5^2$. (d) Haar wavelet $\mathbf{W}_3^2 \otimes \mathbf{V}_5^2$. Note: the gray background indicates zero values, white indicates positive values, and black indicates negative values.

Haar wavelet $\mathbf{V}_m^1 \otimes \mathbf{W}_n^1$ is a 2 by 2 square as well. For the Daubechies wavelets, the supports of the wavelets $\mathbf{V}_m^1 \otimes \mathbf{W}_n^1$ are also small squares, although not 2 by 2 ones. The Daub4 wavelets $\mathbf{V}_m^1 \otimes \mathbf{W}_n^1$, for instance, all have supports in some 4 by 4 square (if we allow for wrap-around at image boundaries).

Similar definitions hold for the other subimages of the 1-level transform. The values of the diagonal fluctuation $\mathbf{d}^1$ are scalar products of the image with the wavelets $\mathbf{W}_m^1 \otimes \mathbf{W}_n^1$, and the values of the vertical fluctuation $\mathbf{v}^1$ are scalar products of the image with the wavelets $\mathbf{W}_m^1 \otimes \mathbf{V}_n^1$. All of these wavelets have energy 1 and average value 0, and supports in small squares. The values of the trend $\mathbf{a}^1$ are scalar products of the image with the scaling images $\mathbf{V}_m^1 \otimes \mathbf{V}_n^1$. Each of these scaling images has energy 1 and average value of 1/2, and support in some small square.

What is true for the first level remains true for all subsequent levels. The values of each subimage $\mathbf{a}^k$, $\mathbf{h}^k$, $\mathbf{d}^k$, and $\mathbf{v}^k$ are computed by scalar products with the scaling images $\mathbf{V}_m^k \otimes \mathbf{V}_n^k$, and the wavelets $\mathbf{V}_m^k \otimes \mathbf{W}_n^k$,

$\mathbf{W}_m^k \otimes \mathbf{W}_n^k$, and $\mathbf{W}_m^k \otimes \mathbf{V}_n^k$, respectively. In Figure 2.16 we show graphs of a 2-level Haar scaling image and three 2-level Haar wavelets. Notice how these images correspond with the subimages of the 2-level Haar transform. For instance, the scaling image $\mathbf{V}_3^2 \otimes \mathbf{V}_5^2$, shown in Figure 2.16(c), is supported on a small 4 by 4 square with a constant value of 1/4, and the average of an image over this square produces the trend value in the $(3, 5)$ position in the trend subimage $\mathbf{a}^2$. Also notice the horizontal orientation of the Haar wavelet in Figure 2.16(a), which is used for computing a value for the horizontal fluctuation $\mathbf{h}^2$. Similarly, the wavelets in Figures 2.16(b) and 2.16(d) have a diagonal orientation and a vertical orientation, respectively, which is consistent with their being used for computing values for the diagonal fluctuation $\mathbf{d}^2$ and vertical fluctuation $\mathbf{v}^2$.

This concludes a very brief outline of the basic features of 2D wavelets and scaling images. Their essential features are very similar to the 1D case; so for reasons of space we shall limit ourselves to this brief treatment. In the next section we shall begin our discussion of applications of 2D wavelet analysis.

2.8 Compression of images

In this section we begin our treatment of image processing applications with a brief treatment of wavelet compression of images. The basic ideas involved are similar to the methods described previously, in Sections 2.4 and 2.5, for compression of 1D signals. We shall concentrate on one representative example of image compression, which should indicate many of the exciting features of wavelet compression techniques, as well as some of the significant challenges in this new field. In the section that follows we shall expand on some of the ideas introduced here by describing a fascinating subfield of image compression: the compression of fingerprint images.

The image that we shall compress is shown in Figure 2.17(a). It is a standard test image, known as *Lena,* which appears frequently in the field of image processing. While one reason for this may be the attractiveness of the image to many of the male workers in the field, another more serious reason is that the *Lena* image contains various combinations of image properties, such as a large number of curved edges and textures in *Lena's* hair and the feathers on her hat, and combinations of light and dark regions of shading in the background and in *Lena's* face.

The *Lena* image is a 512 row by 512 column discrete image. Its values are gray-scale intensity values from 0 to 255—that is, quantized values at 8 bpp—where 0 indicates pure black and 255 indicates pure white, and other values indicate shades of gray between these two extremes. To compress

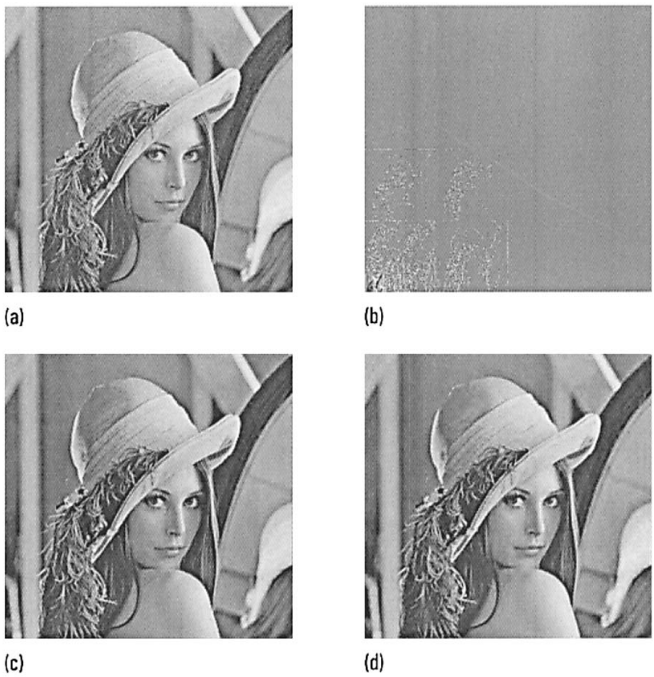

FIGURE 2.17
(a) *Lena* image. (b) 4-level Coif12 transform. (c) Compressed image, 0.43 **bpp. (d) Compressed image,** 0.24 **bpp.**

the *Lena* image we shall use its 4-level Coif12 transform; this transform is shown in Figure 2.17(b). It is clear from this figure that the first level fluctuations—$\mathbf{h}^1$, $\mathbf{d}^1$, and $\mathbf{v}^1$—contain very little energy. An elementary compression could be done by simply omitting all these fluctuations from the compressed version. This is equivalent to compressing the *Lena* image $\mathbf{f}$ to its first level trend $\mathbf{a}^1$, hence compressing *Lena* to 2 bpp.

Much more compression can be obtained, however, if we modify the simple compression described above along the lines of the compression method described for the 1D signal, *greasy,* at the end of Section 2.5. In Figure 2.17(c) we show a compression obtained by quantizing the fourth trend subimage, $\mathbf{a}^4$, of the 4-level Coif12 transform at 8 bpp, and quantizing the second, third, and fourth fluctuations at 7 bpp, and omitting the first fluctuations entirely. Computing separate entropies for each level, we obtain an estimate of 0.43 bpp needed to encode the significant values in the transform. Since the significance map in this case consists of bits of 1's and 0's for only the lower left quadrant of the quantized transform, it accounts for 0.25 bpp without compression of any kind. Examining Figure 2.17(b), however, reveals large areas that consist only of 0's. Therefore, the signif-

icance map can be significantly compressed. The image in Figure 2.17(c) is virtually indistinguishable from the original *Lena* image, even though it has been compressed by roughly 16:1.

Even further compression of the *Lena* image is possible. For instance, if the fourth trend is quantized at 9 bpp and the second, third, and fourth fluctuations are quantized at 6 bpp, then the estimated average number of bits needed to encode the significant transform values is 0.24 bpp. This represents a much greater compression; yet, the compressed image shown in Figure 2.17(d) is still almost indistinguishable from the original. This last statement is certainly true for the printed image shown in the figure; however, when the images are displayed on a computer screen, the 0.24 bpp compressed image exhibits some defects in comparison to the original.

One quantitative measure of how accurately a compressed image $\mathbf{g}$ approximates the original image $\mathbf{f}$ is the *relative* 2*-norm difference* $\mathcal{D}(\mathbf{f}, \mathbf{g})$ defined by

$$\mathcal{D}(\mathbf{f}, \mathbf{g}) = \frac{\sqrt{(f_{1,1} - g_{1,1})^2 + (f_{1,2} - g_{1,2})^2 + \cdots + (f_{N,M} - g_{N,M})^2}}{\sqrt{f_{1,1}^2 + f_{1,2}^2 + \cdots + f_{N,M}^2}}$$

$$= \sqrt{\mathcal{E}_{\mathbf{f}-\mathbf{g}}/\mathcal{E}_{\mathbf{f}}}\,. \tag{2.45}$$

For example, if $\mathbf{f}$ is the *Lena* image and $\mathbf{g}$ is the 0.43 bpp compressed image in Figure 2.17(c), then $\mathcal{D}(\mathbf{f}, \mathbf{g}) = 0.040$. While if $\mathbf{g}$ is the 0.25 bpp compressed image in Figure 2.17(d), then $\mathcal{D}(\mathbf{f}, \mathbf{g}) = 0.046$. This gives quantitative support to the subjective impression that the compressed image in Figure 2.17(c) is a closer approximation to the original image.

As a rule of thumb, *if* $\mathcal{D}(\mathbf{f}, \mathbf{g}) \leq .05$*, then* $\mathbf{g}$ *is an acceptable approximation to* $\mathbf{f}$. This is certainly true for the images in Figure 2.17. The use of $\mathcal{D}(\mathbf{f}, \mathbf{g})$ as a measure of acceptable approximation, however, does not always equate with the perceptions of our visual systems. No such quantitative measure is known, although interesting proposals based on the magnitudes of wavelet transform values have been made in recent papers that we list in the references.

These compressions of the *Lena* image are hardly state of the art. By combining the best available techniques of entropy encoding of the significant quantized transform values with a sophisticated method of compressing the significance map known as *zero-tree encoding,* it is possible to compress *Lena* at a ratio of at least 50:1 without noticeable degradation. The foundation of the zero-tree method is relatively simple. It rests on the fact that trend images are accurate reproductions, at lower resolutions, of the original image. Consequently, in the *Lena* image for instance, regions such as her shoulder which have relatively constant intensity[7] will produce in-

[7] Or for which intensity varies along a flat, linear gradient.

significant values *in the same relative locations* at several consecutive levels. Hence, an insignificant transform value in, say $\mathbf{h}^3$, will often correspond to *four* insignificant values in $\mathbf{h}^2$ in the same relative location (relative to the original image). Likewise, each of those insignificant values in $\mathbf{h}^2$ will often correspond to four more insignificant values, 16 in all, in the same relative locations in $\mathbf{h}^1$. Consequently, these values can be grouped in a data structure, called a *zero-tree*. Encoding these zero-trees with just a single bit is analogous to run-length encoding of zeros, but is much better correlated to the structure of the significance map and produces phenomenal compression ratios. There are very lucid discussions of these matters in some of the original papers which we list in the references. We shall also give some further discussion of zero-trees at the end of the next section.

One aspect of image compression that our brief introduction has omitted is the compression of color images. The way in which most color images are encoded, as Red-Green-Blue (RGB) intensities, makes it possible to consider their compression as a relatively straightforward generalization of compression of gray-scale images. The reason for this is that the total *intensity,* I, which equals the average of the R, G, and B intensities, has much more effect on the visual perception of the color image than the two color values of *hue,* H, and *saturation,* S. Consequently, the RGB image is mapped to an IHS image before compression is done. (The formulas for this mapping are described in the references for this chapter.) After performing this mapping the I, H, and S values are compressed as separate images, using the methods for gray-scale compression illustrated above. Much greater compression can be done on the H and S images than on the I image because of the much lower sensitivity that the human visual system has for variations in hue and saturation. More details on color compression can be found in the references for this chapter.

2.9 Fingerprint compression

Fingerprint compression is an interesting case of image compression. In this section we shall briefly outline the essential ideas underlying wavelet compression of fingerprints. A wavelet-based compression algorithm has been adopted by the U.S. government, in particular by the FBI, as its standard for transmitting and storing digitized fingerprints.

A *fingerprint image* is an image of a fingerprint in an 8-bit gray-scale format. A typical fingerprint record—consisting of ten fingerprints plus two additional thumbprints and two full handprints—when digitized as images produces about 10 megabytes of data. This magnitude of data poses significant problems for transmission and storage. For example, to transmit

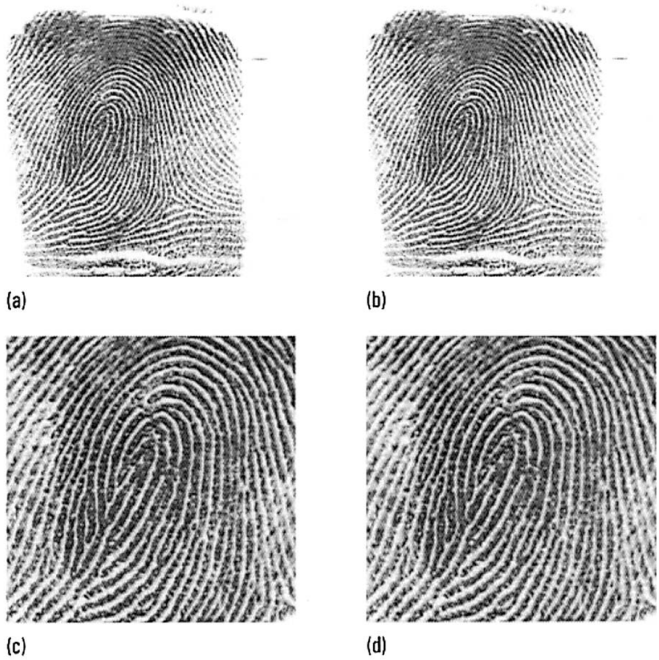

FIGURE 2.18
(a) Fingerprint 1, 8 bpp. (b) Compressed version, 0.77 bpp. (c) Central whorl in (a). (d) Central whorl in (b).

one fingerprint record over a modem, say at 28,000 bits per second with an overhead of around 20%, would require about 1 hour. This is painfully slow when identification needs to be done quickly, as is often the case in criminal investigations. If fingerprint images could be compressed at, say, a 20:1 ratio *without noticeable loss of detail,* then the transmission of one fingerprint record could be reduced to just 3 minutes. This would greatly facilitate the large number of background checks (around 30,000 each day) that are needed.

Besides the transmission problem, there are also problems stemming from the gargantuan magnitude of the FBI's fingerprint archive. This archive contains over 25 million records. Digitizing these records at 8 bpp would produce over 250 trillion bytes of data! Compressing each image by a factor of 20:1 would greatly ease the storage burden for an archive of these fingerprint images.

The wavelet-based compression algorithm adopted by the U.S. government is called the *Wavelet/Scalar Quantization* (WSQ) method. We will not try to give a complete description of the WSQ method. Rather, we shall describe enough of its essential aspects, so that the reader should be

able to confidently read some of the excellent articles by the creators of the WSQ method. These articles are listed in the Notes and References section at the end of this chapter.

In this section, we illustrate the gist of the WSQ method by discussing an example of a fingerprint compression using a wavelet transform. This initial example will be expanded in Chapter 4, in order to give further insight into the WSQ method.

Consider the test image, *Fingerprint 1,* shown in Figure 2.18(a). To compress this image we used a 4-level Coif18 transform. The fourth trend $\mathbf{a}^4$ was quantized at 9 bpp and the various fluctuations were quantized at 6 bpp. This produces an estimated 0.77 bpp needed to encode the significant transform values. Compared with 8 bpp needed for the original image, this represents roughly 10:1 compression. Here we are ignoring the bits needed to encode the significance map; we shall give some justification for doing so at the end of this section. The compressed image is shown in Figure 2.18(b). This image is virtually indistinguishable from the original image in (a).

As a quantitative measure of the accuracy of the compressed image, we calculate the relative 2-norm difference $\mathcal{D}(\mathbf{f}, \mathbf{g})$ defined in the previous section. If $\mathbf{f}$ is the Fingerprint 1 image and $\mathbf{g}$ is the compressed image, then $\mathcal{D}(\mathbf{f}, \mathbf{g}) = .035$. This value is significantly less than the accuracy value of .05 proposed in the previous section.

Often portions of fingerprints need to be magnified in order to compare certain details, such as *whorls.* In Figures 2.18(c) and (d) we show magnifications of the central whorl from Fingerprint 1 and from its compressed version. The compressed version is still virtually indistinguishable from the original in these magnifications. For these magnifications the relative 2-norm difference is .054, which is very near to the accuracy value of .05. It is interesting to note that even the tiny indentations in the middle of the fingerprint ridges, which correspond to sweat pores, are accurately preserved by the compressed image. The locations of these sweat pores are, in fact, legally admissible for identification purposes.

While this example illustrates that a wavelet transform compression of fingerprints can work reasonably well, we can do even better. In Chapter 4, we will show that a *wavelet packet* transform of Fingerprint 1 produces a significantly greater compression. The WSQ method is essentially just a slight modification of this wavelet packet method.

Remarks on the significance map

In our discussion above of a wavelet compression of Fingerprint 1, we ignored the presence of the significance map. In this subsection we will explain why the significance map can be greatly compressed, and consequently does not contribute much to the number of bits needed to describe

the compressed fingerprint image.

We begin by noting that only 12.7% or about one eighth of all of the transform values for Fingerprint 1 are significant. Hence there will be a large preponderance of zero bits in the significance map, which implies that it should be very compressible.

More important, however, many of these zero bits are arranged in the zero-tree data structures mentioned at the end of the previous section. To see this, we show the significance map for the compression of Fingerprint 1 in Figure 2.19. A careful inspection of this significance map reveals many such zero-trees just by sight alone. We start by locating the significance bits for the fourth trend $\mathbf{a}^4$ as the small white square in the lower left corner of Figure 2.19. Directly above the top right corner of this square is a sequence of several gray blobs; these represent clusters of zero bits at the right edge of the fourth horizontal fluctuation $\mathbf{h}^4$. Corresponding to each one of these zero bits are four zero bits, in the same relative location in the third horizontal fluctuation $\mathbf{h}^3$. Thinking of this as an expansion of the area of the blobs by a factor of four, we can see that there is room for these larger blobs lying within a gray rectangle at the edge of the third horizontal fluctuation. Moreover, this entire gray rectangle expands by a factor of four to produce another gray rectangle lying along the side of the second horizontal fluctuation, and this second gray rectangle expands by a factor of four to produce a third gray rectangle along the side of the first horizontal fluctuation.

We have thus found that each zero bit in the original set of gray blobs lies at the root of a four-level zero-tree composed of 85 zero bits. Each of these zero-trees can be encoded with one symbol, say $\mathcal{Z}$, which is in effect an 85:1 compression of each of these zero-trees. Furthermore, each of the extra zero bits within the gray rectangle at the side of the third horizontal fluctuation (the ones not contained within the four-level zero-trees) lies at the root of a three-level zero-tree composed of 20 zero bits. Each of these zero-trees can be collapsed to a symbol $\mathcal{Z}$, producing in effect a 20:1 compression of these trees.

The reader might find it amusing to visually locate other zero-trees. They make up a substantial percentage of all the zero bits in the significance map, and allow for high compressibility.

Finally, we note that the method of zero-tree compression encodes the signs of the significant transform values as symbols (say $\mathcal{P}$ and $\mathcal{N}$) within the significance map. That is, instead of 1 for a significant transform value, a symbol of $\mathcal{P}$ for positive or $\mathcal{N}$ for negative is used instead. Essentially no extra bits are needed for this, because the sign bits can be dropped from the encoding of the significant transform values, leaving only their absolute values to be encoded. Often this significantly decreases the number of bits needed for encoding the significant transform values. For example, for Fingerprint 1, if only the absolute values of the significant 4-level Coif18

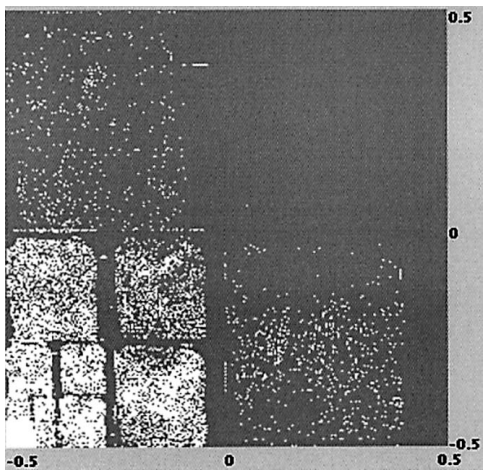

FIGURE 2.19
Significance map for compressed fingerprint. Gray pixels are zero bits, white zeros are one bits.

transform values are encoded, then only 0.64 bpp are needed. This is a 17% reduction from the 0.77 bpp needed when sign bits were included.

For the reasons described above, the significance map often compresses to such a large degree that (taking into account the decrease of bits needed when sign bits are ignored) we can safely ignore it when making a rough estimate of the bpp needed in a wavelet compression of fingerprints.

2.10 Denoising images

In this section we shall describe some fundamental wavelet based techniques for removing noise from images.[8] Noise removal is an essential element of image processing. One reason for this is that many images are acquired under less than ideal conditions and consequently are contaminated by significant amounts of noise. This is the case, for example, with many medical images. Another reason is that several important image processing operations, such as contrast enhancement, histogram equalization,

[8]Adopting audio terminology, undesired changes in image values are called *noise*.

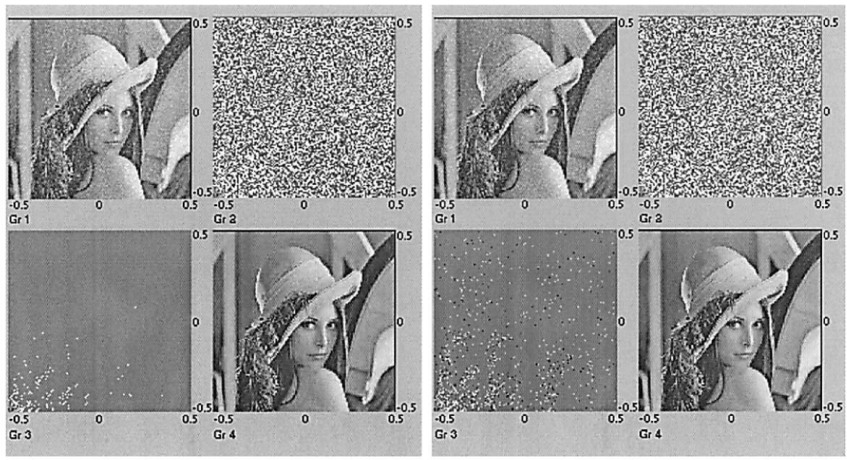

FIGURE 2.20
(a) Hard threshold denoising: Gr 1, *noisy Lena;* Gr 2, 4-level Coif12 transform; Gr 3, hard thresholded transform; Gr 4, denoised image. (b) Soft threshold denoising; the soft thresholded transform is Gr 3.

and edge enhancement, work much better if random noise is absent.

The basic concepts of wavelet denoising of images are similar to those described previously for 1D audio signals in Section 2.6. By choosing a threshold that is a sufficiently large multiple of the standard deviation of the random noise, it is possible to remove most of the noise by thresholding of wavelet transform values. We shall examine how this procedure, which is known as *hard thresholding,* works on a noisy version of the *Lena* image. A simple modification of this method, known as *soft thresholding,* will also be introduced and compared with hard thresholding. We shall then conclude the section with a brief discussion of further aspects of wavelet denoising, which should indicate some of the variety of techniques that wavelet analysis provides for denoising images.

Our first example of image denoising is a hard threshold denoising of a noisy version of the *Lena* image. This image, which we shall refer to as *noisy Lena,* is shown at the top left of Figure 2.20(a). It was obtained by adding Gaussian random noise to the *Lena* image, shown in Figure 2.17(a).

We saw in the previous section that the *Lena* image can be effectively compressed using a 4-level Coif12 transform. This was because there were relatively few, high-energy transform values which captured most of the energy of the image. Therefore, a 4-level Coif12 transform should be an effective tool for denoising the *noisy Lena* image. At the upper right of

Figure 2.20(a) we show a 4-level Coif12 transform of the *noisy Lena* image. An estimate was made of the standard deviation σ of the noise values, using values in the central portion of the diagonal fluctuation $\mathbf{d}^1$. The threshold value T was then set at 4.5σ as per Formula (2.38). At the lower left of Figure 2.20(a) we show the hard thresholded transform; all transform values whose magnitudes are less than T are set equal to 0, while the remaining transform values are retained as significant values. An inverse transform was performed on this thresholded transform, producing the denoised image at the lower right in Figure 2.20(a).

The denoised *Lena* image is clearly an improvement over the noisy version. The RMS Error between the denoised image and the uncontaminated *Lena* image is 15.8. This shows a small reduction from the RMS Error of 17.8 for the *noisy Lena* image, but the amount of reduction does not seem to be large enough to accurately reflect the perceived reduction of noise.[9] Later in this section, we shall discuss an alternative measure which better reflects the amount of noise reduction.

We shall now describe the soft thresholding method of denoising. Soft thresholding is a simple modification of hard thresholding. Hard thresholding consists of applying the following function

$$H(x) = \begin{cases} x & \text{if } |x| \geq T \\ 0 & \text{if } |x| < T \end{cases} \tag{2.46}$$

to the wavelet transform values. See Figure 2.21(a). As can be seen in this figure, the hard threshold function H is not continuous, and thus greatly exaggerates small differences in transform values whose magnitudes are near the threshold value T. If a value's magnitude is only slightly less than T, then this value is set equal to 0, while a value whose magnitude is only slightly greater than T is left unchanged. Soft thresholding replaces the discontinuous function H by a continuous function S, such as[10]

$$S(x) = \begin{cases} x & \text{if } |x| \geq T \\ 2x - T & \text{if } T/2 \leq x < T \\ T + 2x & \text{if } -T < x \leq -T/2 \\ 0 & \text{if } |x| < T/2. \end{cases} \tag{2.47}$$

See Figure 2.21(b). This soft threshold function S does not exaggerate the gap between significant and insignificant transform values.

In Figure 2.20(b) we show a soft threshold denoising of the *noisy Lena* image, using a 4-level Coif12 transform, and also using the same threshold value T as for the hard thresholding in Figure 2.21(a). The soft thresholded transform is shown at the lower left of Figure 2.18(b). It can be

[9]This may not be apparent from the printed images, in which case we urge the reader to examine these images on a computer display using FAWAV.

[10]The function in (2.47) is not the only soft threshold function that is used.

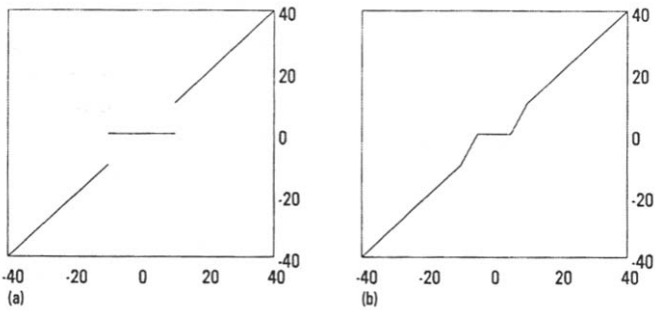

FIGURE 2.21
(a) Hard threshold function, $T = 10$. (b) Soft threshold function, $T = 10$.

clearly seen that the soft thresholding retains more transform values than the hard thresholding. Applying the inverse transform to the soft thresholded transform produces the denoised image at the lower right of Figure 2.21(b). Although it is difficult to distinguish the printed versions of the two denoised images, the computer displayed versions—which are accessible from the figure files available from the FAWAV website—are clearly distinguishable, with the soft threshold denoising appearing to be slightly superior. The RMS Error confirms this subjective judgment. For the soft threshold denoising the RMS Error is 15.20, which is slightly less than the RMS Error of 15.80 for the hard threshold denoising.

Quantitative measures of error

There are several ways of measuring the amount of error between a noisy image and the original image. All of these measures aim to provide quantitative evidence for the effectiveness of noise removal.

One measure is RMS Error, which we used above. Unfortunately, it did not seem to accurately reflect the perceived substantial decrease in noise in either the hard threshold or soft threshold denoising of the *noisy Lena* image.

Another measure is the relative 2-norm difference, $\mathcal{D}(\mathbf{f}, \mathbf{g})$, defined in Formula (2.45). Here $\mathbf{f}$ is the original, uncontaminated image and $\mathbf{g}$ is a noisy image. The errors calculated using $\mathcal{D}$ with the *noisy Lena* example are summarized in Table 2.8. Again, as with RMS Error, this measure of error $\mathcal{D}$ does not seem to accurately reflect the amount of perceived denoising. These two measures, RMS Error and relative 2-norm difference, provide essentially the same information. In fact, simple algebra shows that the ratios of different errors—which are computed in order to compare percentages of denoising—are always the same for these two measures.

A third measure, commonly used in image processing, is *Signal to Noise Ratio* (SNR). If $\mathbf{f}$ is the original image and $\mathbf{g}$ is a noisy image, then the SNR measure of error is defined by

$$\begin{aligned} SNR &= 20\log_{10}\left[1/\mathcal{D}(\mathbf{f},\mathbf{g})\right] \\ &= 10\log_{10}\left[\mathcal{E}_{\mathbf{f}}/\mathcal{E}_{\mathbf{g}-\mathbf{f}}\right]. \end{aligned} \tag{2.48}$$

A rationale for using SNR is that human visual systems respond logarithmically to light energy. The results of applying the SNR measure to our denoising of *noisy Lena* are summarized in Table 2.8. Unlike our other measures, an increase in SNR represents a decrease in error. But, as with the other measures discussed so far, the SNR also does not seem to accurately quantify the amount of decrease in noise in the two denoisings of *noisy Lena.*

The measures of error discussed above have been used for many years. Their deficiencies in relation to accurate quantification of the perceptions of our visual systems are well known. It is generally recognized that they have remained in use, despite their deficiencies, mostly because they have fit well into the type of mathematics used in image processing, making theoretical predictions concerning their values relatively easy to obtain. However, because of the recent explosion of applications of wavelet analysis, it follows that we should also use some measure of error that fits well into a wavelet analysis framework. One such measure, which we shall denote by $\|\mathbf{f}-\mathbf{g}\|_{\mathrm{w}}$, is defined as follows. Suppose that $\{\widehat{f}_{j,k}\}$ are the wavelet transform values for the image $\mathbf{f}$ using one of the Daubechies wavelet transforms, and suppose that $\{\widehat{g}_{j,k}\}$ are transform values for the image $\mathbf{g}$ using the same wavelet transform. The quantity $\|\mathbf{f}-\mathbf{g}\|_{\mathrm{w}}$ is then defined by

$$\|\mathbf{f}-\mathbf{g}\|_{\mathrm{w}} = \frac{|\widehat{f}_{1,1}-\widehat{g}_{1,1}| + |\widehat{f}_{1,2}-\widehat{g}_{1,2}| + \cdots + |\widehat{f}_{N,M}-\widehat{g}_{N,M}|}{MN}.$$

For the *noisy Lena* image we used a 9-level[11] Coif12 transform to compute $\|\mathbf{f}-\mathbf{g}\|_{\mathrm{w}}$. The results are shown in Table 2.8. Notice that this new, wavelet based measure of error finally produces results that more effectively match our perceptions of the success of these denoisings. For example, for the soft thresholding denoising, this new measure shows that there is almost a 50% noise reduction. This seems a more accurate reflection of our perception of the improved quality of the denoised image, certainly more accurate than the reductions of around 10% for the other measures. Of course, one example does not prove the worth of this new measure of error. It does, however, provide a stimulus for further investigation. The reader is invited to carry out more denoisings of the noisy images found at the FAWAV website and compare the various error measures.

[11] Nine levels is the maximum number of levels possible for a 512 by 512 image.

Table 2.8 Error measurements for Figure 2.20

Image	*RMS*	$\mathcal{D}(\mathbf{f},\mathbf{g})$	*SNR*	$\|\mathbf{f}-\mathbf{g}\|_{\mathrm{w}}$
noisy image	17.4	0.131	17.8	9.10
hard thresh. denoise	15.8	0.119	18.7	5.00
soft thresh. denoise	15.2	0.115	19.0	4.62

A couple more remarks need to be made about this new wavelet based measure of error. First, it is not the only measure that can be defined using wavelet transform values. For instance, the terms $|\widehat{f}_{j,k} - \widehat{g}_{j,k}|$ can be raised to powers greater than 1, and/or multiplied by weighting factors that vary depending on what level the transform values belong to. These factors could be chosen, for example, to reflect the different sensitivities of the human visual system to different fluctuation levels. Second, it is worthwhile to reflect on some of the reasons why these new wavelet based measures may provide better noise reduction estimates. Some of the deep mathematical reasons are examined in papers that are listed in this chapter's references. Besides pure mathematical reasons, it may also be that the multiresolution analysis and thresholding performed via wavelets is analogous to the threshold processing performed by the networks of neurons in the human brain in order to decompose visual images into multiple resolutions. Some theoretical models have been proposed for understanding human vision that rely on such analogies.[12]

Further aspects of noise removal

We conclude our discussion of wavelet based denoising of images by briefly outlining some other aspects of this fascinating new field. First, it is important to point out that thresholding—of either the hard or soft kind—is just one approach to denoising of images. Another approach is to modify a thresholding procedure by reducing the size of the threshold to allow in more noisy transform values and, more importantly, more image transform values. The image transform values can often be distinguished from the noise values by using the following principle: *When significant values are found at the same relative locations in each fluctuation subimage, then these values are most likely image values.* The rationale behind this principle is that the values from the image are samples of a smooth function, which will be approximated at lower resolution in each trend subimage, thus pro-

[12] Some of the papers that describe wavelet-like models for human vision are listed in the Notes and references for this chapter.

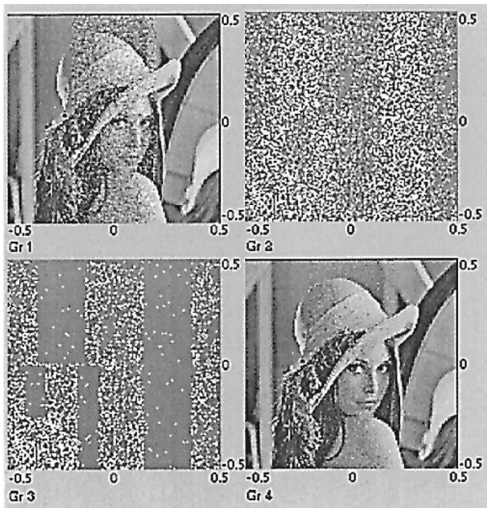

FIGURE 2.22
Removing strip noise: Gr 1, *Lena* with vertical strip noise; 4-level Coif12 transform of noisy image; Gr 3, denoised transform; Gr 4, denoised image.

ducing a hierarchy of similar fluctuation subimages along levels. This can be seen clearly in the octagon image transform in Figure 2.15(d) and in the transform of the *Barb* image in Figure 4.2(a). An application of these ideas to denoising medical images is described in the references.

Second, the fact that most wavelets are supported over small squares implies that wavelet denoising can focus on specific regions of an image where the noise is concentrated. For example, consider the noisy version of the *Lena* image shown at the top left of Figure 2.22. In this image the noise is concentrated along a vertical strip through the middle of the image. Noise analogous to this strip noise occurs, for example, in MRI imaging. Notice that the 4-level Coif12 transform of this noisy image has kept the noise concentrated in vertical strips along the middle of each fluctuation subimage. This is a consequence of the localized support of the Coif12 wavelets. By restricting the soft thresholding to just these vertical strips, and not modifying at all the regions outside these strips, it is possible to reduce the noise along the strip in the image while leaving the noise-free regions outside the strip essentially unaltered. At the lower left of Figure 2.22 we show the transform modified by soft thresholding on the strips in the first and second level fluctuations. The inverse transform of this modified transform is at the lower right of Figure 2.22. This denoised image shows some errors, but they are concentrated in the vertical strip where the noise

was. Outside this vertical strip, the denoised image values are unchanged from their original, uncontaminated values. A summary of error measures is shown in Table 2.9. They indicate that this denoising was quite effective. The wavelet based measure, in particular, shows that the denoising reduced the noise by close to a factor of 3.

Table 2.9 Error measurements for Figure 2.22

Image	*RMS Error*	$\|\mathbf{f}-\mathbf{g}\|_{\mathrm{w}}$
noisy image	15.54	7.66
denoised image	8.59	2.68

As a final example, we show an example of denoising of *multiplicative noise.* This type of noise occurs, for example, with the *speckle noise* that contaminates images produced with laser light. When there is multiplicative noise, the noisy image $\mathbf{f}$ satisfies

$$\mathbf{f} = \mathbf{s}\,\mathbf{n}, \tag{2.49}$$

where $\mathbf{s}$ is the original uncontaminated image and $\mathbf{n}$ is the noise. Equation (2.49) states that the values of $\mathbf{f}$ satisfy $f_{j,k} = s_{j,k}n_{j,k}$. At the left of Figure 2.23 we show a noisy version of the *Lena* image that has been contaminated with multiplicative noise whose values are all positive. Since the values of $\mathbf{f}$ are also positive (except for a few isolated zeros), we can turn this multiplicative noise into additive noise using a logarithm. By applying a base 4 logarithm to the values of the images, we obtain

$$\log_4 \mathbf{f} = \log_4 \mathbf{s} + \log_4 \mathbf{n}. \tag{2.50}$$

Actually, to be precise, a tiny fudge term of .0000001 was added to the values of $\mathbf{f}$ to prevent any logarithms of zero values.

Equation (2.50) shows that the multiplicative noise $\mathbf{n}$ has become an additive noise $\log_4 \mathbf{n}$. Applying a threshold denoising to the image $\log_4 \mathbf{f}$, followed by an application of base 4 exponentiation, we obtained the denoised image shown at the right of Figure 2.23. The summary of error measures shown in Table 2.10 reveals how effective this denoising was. The wavelet based measure, for instance, shows that the noise was reduced by a factor of 3.

We have tried in this section to indicate several of the many features that wavelet based denoising enjoys. Our apologies to the reader if he or she is tired of seeing *Lena.* We concentrated on this particular image in order to reduce the number of figures. At the FAWAV website, we provide many more images on which the reader may test these ideas.

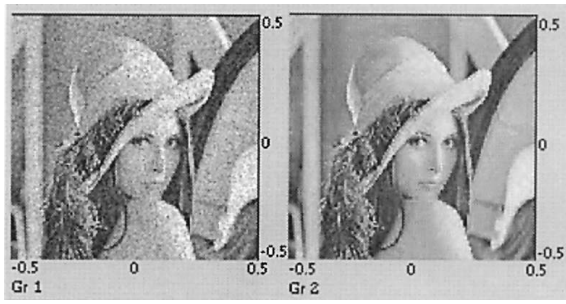

FIGURE 2.23
Removing multiplicative noise: Gr 1, *Lena* with multiplicative random noise; Gr 2, denoised image.

Table 2.10 Error measurements for Figure 2.23

Image	*RMS Error*	$\|\mathbf{f}-\mathbf{g}\|_{\mathrm{w}}$
noisy image	20.21	14.73
denoised image	10.08	4.70

2.11 Some topics in image processing

We conclude our introduction to 2D wavelet analysis with a brief discussion of a few examples of wavelet based techniques in image processing. These examples are meant to provide the reader with a small sampling from the huge variety of applications that have appeared in the last few years. Further examples can be found in the references for this chapter.

Edge detection

One task in image processing is to produce outlines—the edges—of the various objects that compose an image. As shown in Figure 2.15, a wavelet transform can detect edges by producing significant values in fluctuation subimages. One way to produce an image of these edges is to simply perform an inverse transform of only the fluctuation values, after setting all the trend values equal to zero. This produces an image of the first-level detail image $\mathbf{D}^1$. For example, in Figure 2.24(a) we show such a modification of the 1-level Coif6 transform of the octagon image obtained by setting all its first trend values equal to zero. By performing an inverse transform of

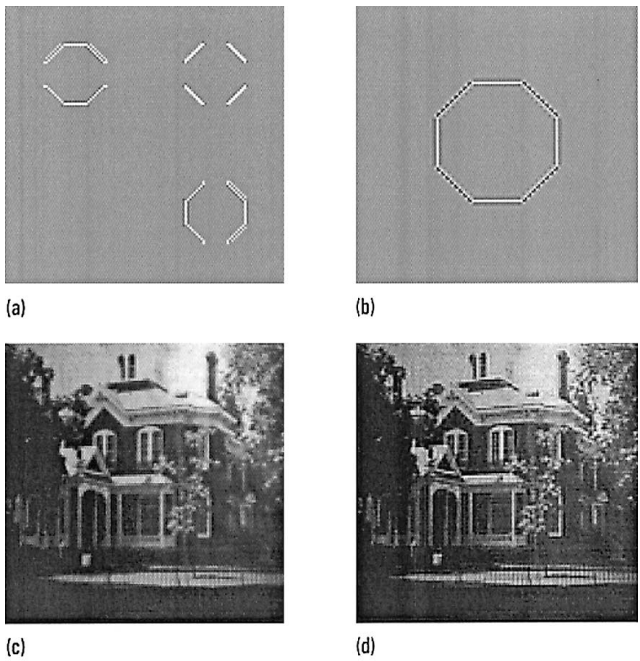

FIGURE 2.24
(a) 1-level Coif6 transform of octagon image with trend subimage values all replaced by zeros. (b) Inverse transform of image in (a). (c) Image of a house. (d) Edge enhanced image.

this modified transform, we produced the detail image $\mathbf{D}^1$ shown in Figure 2.24(b). This figure clearly highlights the edges of the original image. Of course, this image could be further processed to increase the highlighting of just the outline of the octagon. For instance, only the most intense, whitest portion of the image could be retained, using a thresholding operation.

Edge enhancement

If we can detect edges accurately, then we can also enhance their appearance in an image. This will sharpen images that suffer from dull, blurry edges. For example, in Figure 2.24(c) we show an image of a house that suffers from blurred edges. In order to sharpen the edges of this image, we used the following method.

Edge Enhancement Method

Step 1. Perform a wavelet transform of image.

Step 2. Multiply fluctuation values by a constant larger than 1, and leave trend values unchanged.

Step 3. Perform an inverse transform of modified image from Step 2.

To produce the edge enhanced image of the house, we used a 1-level Daub4 transform in Step 1, and we multiplied the first-level fluctuation values by the constant 3 in Step 2.

Comparing the two images of the house in Figure 2.24, we can see that the edge enhanced image is a sharper image than the original. Some details can be seen more clearly in the enhanced image, such as the fencing in front of the house, and the woodwork at the top of the roof. Particularly striking is the fact that in the right central window a vertical window divider has been rendered visible in the enhanced image. This kind of edge sharpening and improvement in detail visibility is clearly of fundamental importance in medical and biological imaging. The edge enhancement method just described is only the simplest of many related procedures. For instance, if a multiple level transform is performed in Step 1, then Step 2 can be done differently by using different constants for each fluctuation level.

Image recognition

Teaching a machine to recognize images is a very difficult problem. For example, we might want to program a computer to recognize images of human faces. Image recognition is important in areas such as data retrieval, object identification, and the science of human vision. *Data retrieval* refers to finding a match to a given image from a huge archive of images and then retrieving more data associated with that image (e.g., name, date of birth, etc.). *Object identification* refers to the problem of identifying the location, or lack thereof, of a particular object such as an aircraft within a complicated scene. Image retrieval relates to some aspects of the *science of human vision* because of our desire to know how our brains recognize images out of the data supplied by our eyes. While programming a computer to recognize an image is not the same as understanding how our brains do it, nevertheless, insights gained from the one task will certainly provide ideas for the other.

In Figure 2.25 we illustrate an elementary problem of image recognition. Suppose that we want a machine to identify which of the four images shown in Figure 2.25(a) is the best match to the denoised version of *Lena* in Figure 2.20(a). We choose to look for a match to the *denoised Lena,* rather than *Lena,* since it is more likely that we would not receive a precise image for matching, but rather a noisy version. It is also likely that the image we wish to match differs in other ways (different vantage point, different facial expression, etc.) from the image of that person in the archive. This raises the considerably more difficult problem of matching distinct images

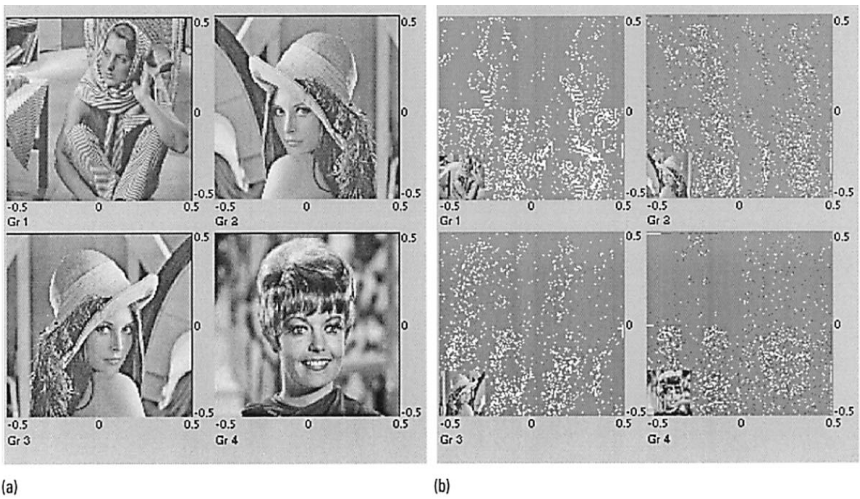

FIGURE 2.25
(a) Four images. (b) 2-level Coif18 transforms of these images.

of the same person. It turns out that the wavelet based algorithm for image matching outlined below helps with this problem, too.

A simple method for matching would be to compute an error measure, say $\mathcal{D}(\mathbf{f}, \mathbf{g})$, where $\mathbf{f}$ is the denoised *Lena* image and $\mathbf{g}$ is each of the images in our archive. For the four images in Figure 2.25(a) this method works well; as shown in the column labeled *Full* in Table 2.11, the image labeled Gr 3 produces $\mathcal{D}(\mathbf{f}, \mathbf{g}) = 0.066$, and all other images produce $\mathcal{D}(\mathbf{f}, \mathbf{g}) > 0.48$. So the image in Gr 3 is clearly the best match. There is a significant problem with this approach, however, since an actual archive will certainly contain far more than four images—it might contain tens of thousands of images. The images we are using are 512 by 512, hence each image contains over a quarter of a million values. The time delay—due to the enormous number of calculations—associated with finding $\mathcal{D}(\mathbf{f}, \mathbf{g})$ for tens of thousands of images is prohibitively large.

Wavelet analysis provides a flexible approach for significantly reducing the number of computations needed for this image matching problem. The wavelet based approach also incorporates its own compression procedure, which allows for the archive to be stored in a compressed format. To illustrate this wavelet approach to image matching, consider the 2-level Coif18 transforms of the four images in our small archive shown in Figure 2.25(b). Notice that our visual systems can easily distinguish the four images based only on their second trend subimages, which are one-sixteenth the size of the original image. These smaller images are called *thumbnail* images. If

we simply compute the error measures for these second trends only, then we obtain the results shown in the column labeled *Second* in Table 2.11. This new computation easily matches the *denoised Lena* image with the correct image of *Lena* in our tiny archive. Since the second trends are one-sixteenth the size of the original images, this new computation is sixteen times faster than the first one involving the full images.

Table 2.11 Error measures for images

Image	*Full*	*Second*	*Third*	*Fourth*	*Fifth*
Gr 1	0.481	0.495	0.473	0.459	0.426
Gr 2	0.485	0.491	0.475	0.457	0.410
Gr 3	0.066	0.076	0.063	0.063	0.063
Gr 4	0.546	0.549	0.540	0.529	0.501

What works well for second-level trends also works well for third, fourth, and even fifth trends. As shown in Table 2.11, computing errors for any of these trends clearly determines the correct match. If we use fifth trends to perform error computations, then our calculations proceed over 1000 times faster than if we were to use the full images. This improved performance factor makes it quite practical to search through an archive of tens of thousands of images. It is important to note that in order to achieve this rapid performance, the images must be stored in the archive in their wavelet transformed format. Fortunately, this also allows for the images to be stored in a compressed format.

Another advantage of comparing these very low resolution, fifth trend images is the following. If the given image is only a fair approximation, rather than a perfect replica, of its version in the archive, then the low resolution fifth trend versions should still produce smaller error measures than the fifth trends of very different looking people. This indicates how a wavelet based algorithm facilitates the matching of distinct images of the same person.

Of course, if we are using fifth trends it is quite possible that several images of different people might produce errors that are less than whatever error threshold we have established for matching. In that case, by using the fourth-level fluctuations, we can rapidly construct the fourth-level trend subimages for the subset of images which survived the fifth-level matching test. The error measures can then be computed for these fourth-level trends. Or, better yet, we could instead bypass the reconstruction of the fourth-level trends, and for each image add on an error term computed from the fourth-level fluctuations to the error already computed for the fifth-level trend. This is equivalent to computing errors for the fourth-level trends,

and is a faster computation. Although these additional calculations require more time, it is very likely that the subset of images that survived the fifth trend comparison is much less numerous than the original set. The recursive structure of wavelet transforms enables the implementation of a recursive matching algorithm that successively winnows the subset of possible matching images, using extremely small numbers of computations to distinguish grossly dissimilar images, while reserving the most numerous computations only for a very small number of quite similar images.

This brief outline of an image retrieval algorithm should indicate some of the fundamental advantages that a wavelet based approach provides. One of the most important features of the approach we described is its use of the multiresolution structure (the recursive structure of the trend subimages) to facilitate computations. This has applications to many other problems. For instance, in object identification, a method known as *correlation,* which we will describe in the next chapter, is a computationally intensive method for locating a given object within a complicated scene. Using correlation on third trends instead of the full images, however, can greatly speed up the approximate location of an object.

2.12 Notes and references

Descriptions of the many other types of wavelets—biorthogonal wavelets, spline wavelets, etc.—can be found in [BGG], [DAU], [MAL], [REW], or [STN]. The book [CHU] is particularly good on the topic of spline wavelets. Two good references on digital image processing are [JAH] and [RUS]. The book [ALU] contains several excellent papers on wavelets and their applications in biology and medicine.

Further discussions of quantization can be obtained from [MAL], [VEK], or [DVN]. Some good references on information theory are [ASH], [COT], and [HHJ].

An excellent summary of wavelet-based image compression can be found in [DVN]. A good site for downloading images is located at:

http://links.uwaterloo.ca/bragzone.base.html

Details on the best image compression ratios yet obtained can be viewed at the following website:

http://www.icsl.ucla.edu/~ipl/psnr_results.html

A basic image compression software package can be downloaded from the website:

http://www.cs.dartmouth.edu/~gdavis/wavelet/wavelet.html

This site also has a number of useful links to other sites.

The zero-tree method of image compression is described in the papers [SHA] and [SAP]. Software that implements the zero-tree method can be obtained by FTP from the following address:

ftp://ipl.rpi.pub/EW_Code

or can be downloaded from the following website:

http://www.ipl.rpi.edu/SPIHT

There is an interesting analysis of tree structures in wavelet-based image compression and their relationship with fractal image compression in [DVS].

Formulas for the RGB to IHS mapping can be found in [RUS]. The differences in sensitivity of the human visual system to I versus H and S can be found in [WAN]. The book [WAN] is the best elementary introduction to the science of the human visual system. In [WAN] there is a good summary of MRA and its relation to vision. Wavelet-like models of the human visual system are described in [WAT], [FI1], and [FI2].

The best discussions of fingerprint compression can be found in the articles [BBH] and [BRS] by the creators of the WSQ method. The website:

http://www.c3.lanl.gov/~brislawn

may also be of interest. One of the fingerprints, Fingerprint 8, at the FAWAV website is a cropped version of a fingerprint found at this site. It was supplied to the author courtesy of Dr. Brislawn. The other fingerprints, including Fingerprint 1 in Figure 2.18(a), were obtained from the NIST database at the ftp site:

ftp://sequoyah.nist.gov

Introductory treatments of noise can be found in [BAS] and [FAN]. The method of choosing denoising thresholds for wavelet transforms has received a very complete analysis in the papers [DOJ] and [DJK], and is given a good summary in [MAL]. There is also an excellent survey of various wavelet-based noise removal techniques in [DON]. The use of different thresholds for each fluctuation subimage is utilized in the program SURESHRINK, which can be downloaded from the following website:

http://www.cs.scarolina.edu/ABOUT_US/Faculty/Hilton/shrink-demo.html

Another commonly used measure of errors in images is PSNR, which is closely related to SNR; see [VEK] for the definition of PSNR. A discussion of new wavelet-based measures of errors in images is given in [CDL]. Removing

noise in medical images, based on thresholding and based on the statistics of significant values between levels, is described in [MLF] and [XWH].

There is an interesting discussion in [WI2] of a wavelet based solution of the *rogues gallery problem,* the problem of retrieving an image from a large archive. This discussion differs somewhat from the one given here.