2

HAAR: THE SIMPLEST WAVELET BASIS

1. The one-dimensional Haar wavelet transform — 2. One-dimensional Haar basis functions — 3. Orthogonality and normalization — 4. Wavelet compression

The Haar basis is the simplest wavelet basis. In this chapter, we will begin by examining how a one-dimensional function can be decomposed using Haar wavelets. We will then look at the Haar basis functions in detail and see how Haar wavelet decomposition can be used for compression. Later, in the following three chapters, we'll explore some applications of the Haar basis: image compression, image editing, and image querying.

2.1 The one-dimensional Haar wavelet transform

To get a sense for how wavelets work, let's start out with a simple example. Suppose we are given a one-dimensional "image" with a resolution of 4 pixels, having the following pixel values:

$$[9\ 7\ 3\ 5]$$

This image can be represented in the Haar basis, the simplest wavelet basis, by computing a wavelet transform as follows. Start by averaging the pixels together, pairwise, to get a new lower-resolution image with these pixel values:

$$[8\ 4]$$

Clearly, some information has been lost in this averaging and down-sampling process. In order to be able to recover the original four pixel values from the two averaged pixels, we need to store some *detail coefficients*, which capture that missing information. In our example, we will choose 1 for the first detail coefficient, since the average we computed is 1 less than 9 and 1 more than 7. This single number allows us to recover the first two pixels of our original four-pixel image. Similarly, the second detail coefficient is -1, since $4+(-1)=3$ and $4-(-1)=5$.

Summarizing, we have so far decomposed the original image into a lower-resolution (two-pixel) version and detail coefficients as follows:

Resolution	Averages	Detail Coefficients
4	[9 7 3 5]	
2	[8 4]	[1 −1]

Repeating this process recursively on the averages gives the full decomposition:

Resolution	Averages	Detail Coefficients
4	[9 7 3 5]	
2	[8 4]	[1 −1]
1	[6]	[2]

Finally, we will define the *wavelet transform* (also called the *wavelet decomposition*) of the original four-pixel image to be the single coefficient representing the overall average of the original image, followed by the detail coefficients in order of increasing resolution. Thus, for the one-dimensional Haar basis, the wavelet transform of our original four-pixel image is given by

$$[6\ 2\ 1\ -1]$$

The way we computed the wavelet transform, by recursively averaging and differencing coefficients, is called a *filter bank*—a process we will generalize to other types of wavelets in Chapter 7. Note that no information has been gained or lost by this process: The original image had four coefficients, and so does the transform. Also note that, given the transform, we can reconstruct the image to any resolution by recursively adding and subtracting the detail coefficients from the lower-resolution versions.

Storing the wavelet transform of the image, rather than the image itself, has a number of advantages. One advantage of the wavelet transform is that often a large number of the detail coefficients turn out to be very small in magnitude, as in the larger example of Figure 2.1. Truncating, or removing, these small coefficients from the representation introduces only small errors in the reconstructed image, giving a form of "lossy" image compression. We will discuss this particular application of wavelets in Section 2.4, once we have presented the one-dimensional Haar basis functions.

2.2 One-dimensional Haar basis functions

In the previous section we treated one-dimensional images as sequences of coefficients. Alternatively, we can think of images as piecewise-constant functions on the half-open interval $[0, 1)$. (A *half-open interval* $[a, b)$ contains all values of x in the range $a \le x < b$.) In this new treatment, we will use the concept of a "vector space" from linear algebra. (A refresher on linear algebra can be found in Appendix A.)

A *vector space* V is basically just a collection of "things" (called *vectors*, in this context) for which addition and scalar multiplication are defined. Thus, you can add two vectors, scale a vector by some constant, and so forth. (The full list of axioms can be found in Appendix A.1.)

Until now, we have been thinking of images as sequences of coefficients; let's instead think of them as functions. For example, we can consider a one-pixel image to be a function that is constant over the entire interval $[0, 1)$. Since addition and scalar multiplication of functions are well defined, we can then think of each constant function over the interval $[0, 1)$ as a vector, and we'll let V^0 denote the vector space of all such functions. Similarly, a two-pixel image is a function having two constant pieces over the intervals $[0, 1/2)$ and $[1/2, 1)$. We'll call the space containing all these functions V^1. If we continue in this manner, the space V^j will include all piecewise-constant functions defined on the interval $[0, 1)$ with constant pieces over each of 2^j equal-sized subintervals.

V^{16}

V^8

V^4

V^2

V^1

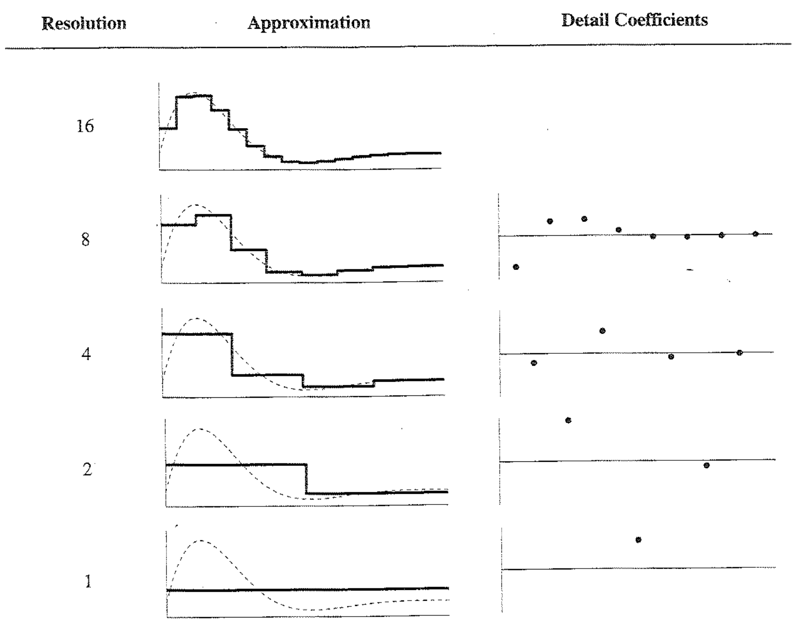


FIGURE 2.1 A sequence of decreasing-resolution approximations to a function (left) along with the detail coefficients required to recapture the finest approximation (right). Note that in regions where the true function is close to being flat, a piecewise-constant approximation is a good one, and so the corresponding detail coefficients are relatively small.

We can now think of every one-dimensional image with 2^j pixels as being an element, or vector, in V^j. Note that because these vectors are all functions defined on the unit interval, every vector in V^j is also contained in V^{j+1}. For example, we can always describe a piecewise-constant function with two intervals as a piecewise-constant function with four intervals, with each interval in the first function corresponding to a pair of intervals in the second. Thus, the spaces V^j are nested; that is,

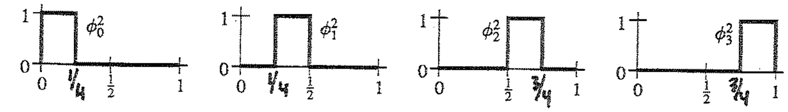


FIGURE 2.2 The box basis for V^2.

$$V^0 \subset V^1 \subset V^2 \subset \cdots$$

This nested set of spaces V^j is a necessary ingredient for the mathematical theory of multiresolution analysis, a topic we will consider more thoroughly in Chapter 7.

Now we need to define a basis for each vector space V^j. (A *basis* for a vector space is defined formally in Appendix A.2. Roughly speaking, a basis consists of a minimum set of vectors from which all other vectors in the vector space can be generated through linear combinations.) The basis functions for the spaces V^j are called *scaling functions* and are usually denoted by the symbol ϕ. A simple basis for V^j is given by the set of scaled and translated *box functions:*

define basis

$$\phi_i^j(x) := \phi(2^j x - i) \qquad i = 0, \ldots, 2^j - 1$$

where

$$\phi(x) := \begin{cases} 1 & \text{for } 0 \le x < 1 \\ 0 & \text{otherwise} \end{cases}$$

As an example, the four box functions forming a basis for V^2 are shown in Figure 2.2.

The *support* of a function refers to the region of the parameter domain over which the function is nonzero. For example, the support of $\phi_0^2(x)$ is $[0, 1/4)$. Functions that are supported over a bounded interval are said to have *compact support.* Note that all of the box functions are compactly supported.

The next step in building a multiresolution analysis is to choose an *inner product* defined on the vector spaces V^j (see Appendix A.3 for a formal definition of inner products). For our running example, the "standard" inner product will do quite well:

build hierachical representation

orthogonal complement

$$\langle f \mid g \rangle := \int_0^1 f(x)\, g(x)\, dx$$

Two vectors u and v are said to be *orthogonal* under a chosen inner product if $\langle u \mid v \rangle = 0$. We can now define a new vector space W^j as the *orthogonal complement* of V^j in V^{j+1}. In other words, W^j is the space of all functions in V^{j+1} that are orthogonal to all functions in V^j under the chosen inner product.

A collection of linearly independent functions $\psi_i^j(x)$ spanning W^j are called *wavelets*. These basis functions have two important properties:

1. The basis functions ψ_i^j of W^j, together with the basis functions ϕ_i^j of V^j, form a basis for V^{j+1}.

2. Every basis function ψ_i^j of W^j is orthogonal to every basis function ϕ_i^j of V^j under the chosen inner product.

important

Remark: Later, in Chapter 7, we'll look at ways in which the definitions of the complement spaces W^j and the wavelets ψ_i^j above can either be made more strict or more relaxed. For example, some authors refer to the functions defined as wavelets above as *pre-wavelets*, reserving the term *wavelets* for functions ψ_i^j that are orthogonal to each other as well. ■

(*) Informally, we can think of the wavelets in W^j as a means of representing the parts of a function in V^{j+1} that cannot be represented in V^j. Thus, the detail coefficients of Section 2.1 are really coefficients of the wavelet basis functions.

The wavelets corresponding to the box basis are known as the *Haar wavelets*, given by

$$\psi_i^j(x) := \psi(2^j x - i) \qquad i = 0, \ldots, 2^j - 1$$

where

$$\psi(x) := \begin{cases} 1 & \text{for } 0 \le x < 1/2 \\ -1 & \text{for } 1/2 \le x < 1 \\ 0 & \text{otherwise} \end{cases}$$

Figure 2.3 shows the two Haar wavelets spanning W^1.

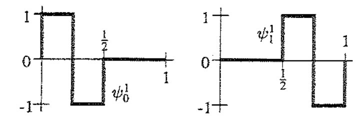


FIGURE 2.3 The Haar wavelets for W^1.

Before going on, let's run through our example from Section 2.1 again, but now applying these more sophisticated ideas. We begin by expressing our original image $\mathcal{I}(x)$ as a linear combination of the box basis functions in V^2:

$$\mathcal{I}(x) = c_0^2\,\phi_0^2(x) + c_1^2\,\phi_1^2(x) + c_2^2\,\phi_2^2(x) + c_3^2\,\phi_3^2(x)$$

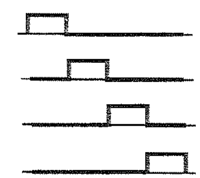
A more graphical representation is

$\mathcal{I}(x) = 9 \times$

$+ 7 \times$

$+ 3 \times$

$+ 5 \times$



resolution 4

Note that the coefficients $c_0^2, \ldots, c_3^2$ are just the four original pixel values [9 7 3 5].

We can rewrite the expression for $\mathcal{I}(x)$ in terms of basis functions in V^1 and W^1, using pairwise $\underbrace{\text{averaging}}_{\text{lowpass}}$ and $\overbrace{\text{differencing}}^{\text{highpass}}$:

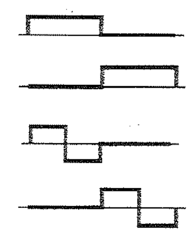
$$\mathcal{I}(x) = c_0^1\,\phi_0^1(x) + c_1^1\,\phi_1^1(x) + d_0^1\,\psi_0^1(x) + d_1^1\,\psi_1^1(x)$$

$= 8 \times$

$+ 4 \times$

$+ 1 \times$

$+ -1 \times$



resolution 2

These four coefficients should look familiar as well.

Finally, we'll rewrite $\mathcal{I}(x)$ as a sum of basis functions in V^0, W^0, and W^1:

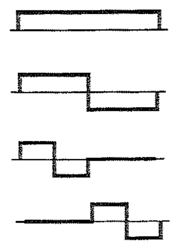
$$\mathcal{I}(x) = c_0^0\,\phi_0^0(x) + d_0^0\,\psi_0^0(x) + d_0^1\,\psi_0^1(x) + d_1^1\,\psi_1^1(x)$$

$= 6 \times$

$+ 2 \times$

$+ 1 \times$

$+ -1 \times$



resolution 1

Once again, these four coefficients are the Haar wavelet transform of the original image. The four functions shown above constitute the Haar basis for V^2. Instead of using the usual four box functions, we can use ϕ_0^0, ψ_0^0, ψ_0^1, and ψ_1^1 to represent the overall average, the broad detail, and the two types of finer detail possible in a function in V^2. The Haar basis for V^j with $j > 2$ includes these four functions as well as even narrower versions of the wavelet $\psi(x)$.

2.3 Orthogonality and normalization

The Haar basis possesses an important property known as *orthogonality*, which is not always shared by other wavelet bases. An orthogonal basis is one in which all of the basis functions, in this case $\phi_0^0, \psi_0^0, \psi_0^1, \psi_1^1, \ldots$, are orthogonal to one another. Note that orthogonality is stronger than the requirement in the definition of wavelets that ψ_i^j be orthogonal only to all scaling functions at the same hierarchy level j.

Another property of some wavelet bases is *normalization*. A basis function $u(x)$ is normalized if $\langle u \mid u \rangle = 1$. We can normalize the Haar basis by replacing our earlier definitions with

$$\phi_i^j(x) := \sqrt{2^j}\,\phi(2^j x - i)$$
$$\psi_i^j(x) := \sqrt{2^j}\,\psi(2^j x - i)$$

where the constant factor $\sqrt{2^j}$ is chosen to satisfy $\langle u \mid u \rangle = 1$ for the standard inner product. With these modified definitions, the new normalized coefficients are obtained by dividing

each old coefficient with superscript j by $\sqrt{2^j}$. Thus, in the example from the previous section, the unnormalized coefficients $[6\ 2\ 1\ -1]$ become the normalized coefficients

$$\left[6\ 2\ \frac{1}{\sqrt{2}}\ \frac{-1}{\sqrt{2}}\right]$$

As an alternative to first computing the unnormalized coefficients and then normalizing them afterwards, we can include normalization in the decomposition algorithm. The following two pseudocode procedures accomplish this normalized decomposition, transforming a set of coefficients in place:

procedure *Decomposition*(c: **array** $[1\,..\,2^j]$ **of reals**)
  $c \leftarrow c/\sqrt{2^j}$ *(normalize input coefficients)*
  $g \leftarrow 2^j$
  while $g \geq 2$ **do**
    DecompositionStep($c[1\,..\,g]$)
    $g \leftarrow g/2$
  end while
end procedure

procedure *DecompositionStep*(c: **array** $[1\,..\,2^j]$ **of reals**)
  for $i \leftarrow 1$ **to** $2^j/2$ **do**
    $c'[i] \leftarrow (c[2i-1] + c[2i])/\sqrt{2}$
    $c'[2^j/2 + i] \leftarrow (c[2i-1] - c[2i])/\sqrt{2}$
  end for
  $c \leftarrow c'$
end procedure

Of course, after we obtain a wavelet decomposition, we need to be able to reconstruct the original data. The following two pseudocode procedures do just that.

procedure *Reconstruction*(c: **array** $[1\,..\,2^j]$ **of reals**)
  $g \leftarrow 2$
  while $g \leq 2^j$ **do**
    ReconstructionStep($c[1\,..\,g]$)
    $g \leftarrow 2g$
  end while
  $c \leftarrow c\sqrt{2^j}$ *(undo normalization)*
end procedure

```
procedure ReconstructionStep(c: array [1 .. 2^j] of reals)
  for i ← 1 to 2^j/2 do
    c'[2i − 1] ← (c[i] + c[2^j/2 + i])/√2
    c'[2i] ← (c[i] − c[2^j/2 + i])/√2
  end for
  c ← c'
end procedure
```

The above pseudocode procedures allow us to work with an *orthonormal* basis: one that is both orthogonal and normalized. As we will see in the next section, using an orthonormal basis turns out to be handy when compressing a function or an image.

2.4 Wavelet compression

The goal of compression is to express an initial set of data using some smaller set of data, either with or without loss of information. For instance, suppose we are given a function $f(x)$ expressed as a weighted sum of basis functions $u_1(x), \dots, u_m(x)$:

$$f(x) = \sum_{i=1}^{m} c_i\, u_i(x)$$

The data set in this case consists of the coefficients $c_1, \dots, c_m$. We would like to find a function approximating $f(x)$ but requiring fewer coefficients, perhaps by using a different basis. That is, given a user-specified error tolerance ε (for lossless compression, $\varepsilon = 0$), we are looking for

$$\hat{f}(x) = \sum_{i=1}^{\hat{m}} \hat{c}_i\, \hat{u}_i(x)$$

such that $\hat{m} < m$ and $\| f(x) - \hat{f}(x) \| \le \varepsilon$ for some norm (see Appendix A.4 for more on norms). In general, one could attempt to construct a set of basis functions $\hat{u}_1, \dots, \hat{u}_{\hat{m}}$ that would provide a good approximation with few coefficients. We will focus instead on the simpler problem of finding a good approximation in a fixed basis. Note that here and elsewhere in the book, when we discuss compression, we are concentrating on reducing the number of coefficients needed to represent a function—and not on the equally challenging problem of *encoding* and storing the necessary information in the fewest possible bits.

One form of the compression problem is to order the coefficients $c_1, \dots, c_m$ so that for every $\hat{m} < m$, the first $\hat{m}$ elements of the sequence give the best approximation $\hat{f}(x)$ to $f(x)$ as measured in the L^2 norm. As we show here, the solution to this problem is straightforward if the basis is orthonormal, as is the case with the normalized Haar basis.

Let $\pi(i)$ be a permutation of $1, \dots, m$ and let $\hat{f}(x)$ be a function that uses the coefficients corresponding to the first $\hat{m}$ numbers of the permutation $\pi(i)$:

$$\hat{f}(x) = \sum_{i=1}^{\hat{m}} c_{\pi(i)}\, u_{\pi(i)}$$

The square of the L^2 error in this approximation is given by

$$\begin{aligned} \| f(x) - \hat{f}(x) \|_2^2 &= \langle f(x) - \hat{f}(x) \mid f(x) - \hat{f}(x) \rangle \\ &= \left\langle \sum_{i=\hat{m}+1}^{m} c_{\pi(i)}\, u_{\pi(i)} \;\middle|\; \sum_{j=\hat{m}+1}^{m} c_{\pi(j)}\, u_{\pi(j)} \right\rangle \\ &= \sum_{i=\hat{m}+1}^{m} \sum_{j=\hat{m}+1}^{m} c_{\pi(i)}\, c_{\pi(j)} \langle u_{\pi(i)} \mid u_{\pi(j)} \rangle \\ &= \sum_{i=\hat{m}+1}^{m} (c_{\pi(i)})^2 \end{aligned}$$

The last step follows from the assumption that the basis is orthonormal, so $\langle u_i \mid u_j \rangle = \delta_{ij}$. The above result indicates that the square of the overall L^2 error is just the sum of the squares of all the coefficients we choose to leave out. We conclude that in order to minimize this error for any given $\hat{m}$, the best choice for $\pi(i)$ is the permutation that sorts the coefficients in order of decreasing magnitude; that is, $\pi(i)$ satisfies

$$|c_{\pi(1)}| \ge \cdots \ge |c_{\pi(m)}|$$

Figure 2.1 demonstrated how a one-dimensional function could be transformed into coefficients representing the function's overall average and various resolutions of detail. Now we

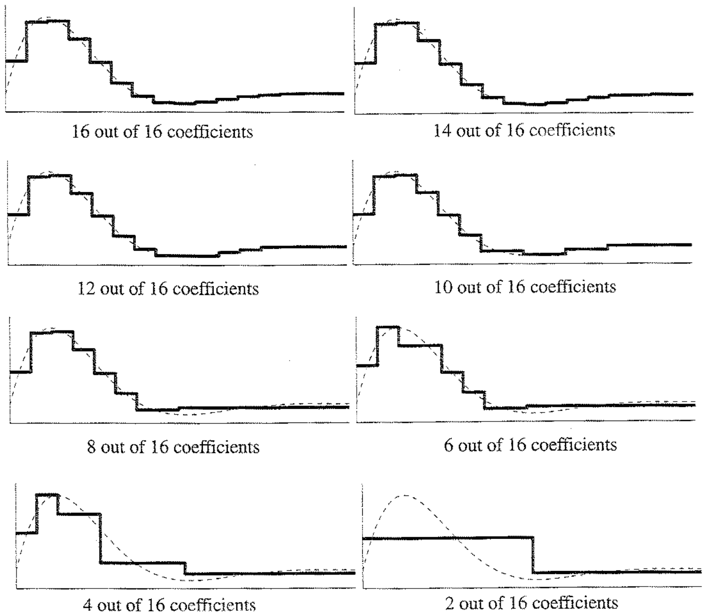


FIGURE 2.4 Coarse approximations to a function obtained using L^2 compression: detail coefficients are removed in order of increasing magnitude.

repeat the process, this time using normalized Haar basis functions. We can apply L^2 compression to the resulting coefficients simply by removing or ignoring the coefficients with smallest magnitude. By varying the amount of compression, we obtain a sequence of approximations to the original function, as shown in Figure 2.4.

3 IMAGE COMPRESSION

1. Two-dimensional Haar wavelet transforms — 2. Two-dimensional Haar basis functions — 3. Wavelet image compression — 4. Color images — 5. Summary

In preparation for image compression, we need to generalize Haar wavelets to two dimensions. First, we consider how to perform a wavelet decomposition of the pixel values in a two-dimensional image. We then describe the scaling functions and wavelets that form a two-dimensional wavelet basis. These tools will enable us to describe image compression as an application of wavelets.

3.1 Two-dimensional Haar wavelet transforms

There are two common ways in which wavelets can be used to transform the pixel values within an image. Each of these transformations is a two-dimensional generalization of the one-dimensional wavelet transform described in Section 2.1.

The first transform is called the *standard decomposition* [5]. To obtain the standard decomposition of an image, we first apply the one-dimensional wavelet transform to each row of pixel values. This operation gives us an average value along with detail coefficients for each row. Next, we treat these transformed rows as if they were themselves an image and apply the one-dimensional transform to each column. The resulting values are all detail coefficients

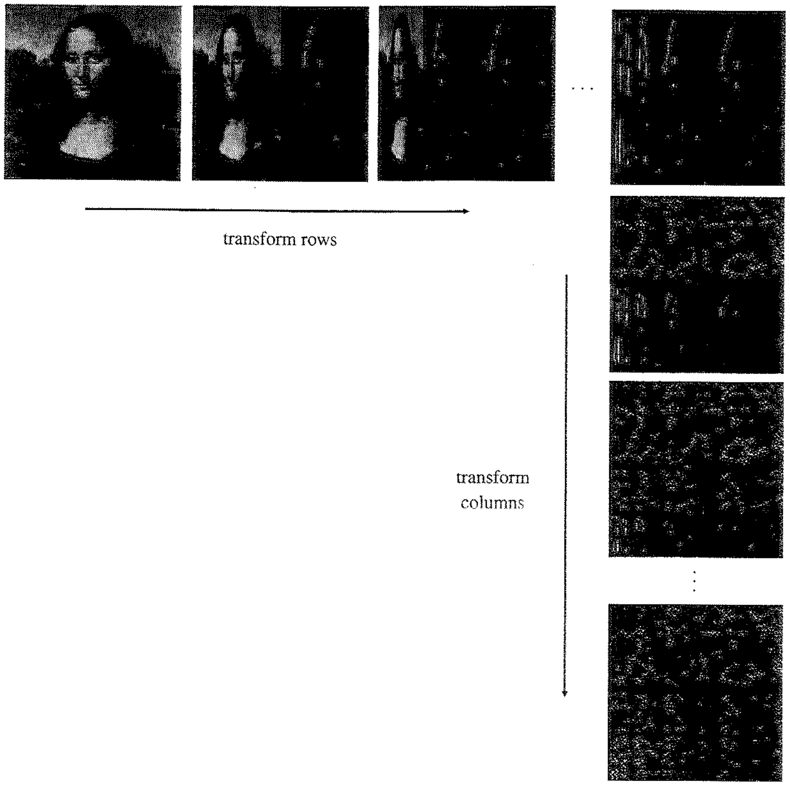


FIGURE 3.1 Standard decomposition of an image.

except for a single overall average coefficient. An algorithm to compute the standard decomposition is given below. Each step of its operation is illustrated in Figure 3.1.

```
procedure StandardDecomposition(c: array [1 .. 2^j, 1 .. 2^k] of reals)
  for row ← 1 to 2^j do
    Decomposition(c[row, 1 .. 2^k])
  end for
  for col ← 1 to 2^k do
    Decomposition(c[1 .. 2^j, col])
  end for
end procedure
```


The corresponding reconstruction algorithm simply reverses the steps performed during decomposition:

```
procedure StandardReconstruction(c: array [1 .. 2^j, 1 .. 2^k] of reals)
  for col ← 1 to 2^k do
    Reconstruction(c[1 .. 2^j, col])
  end for
  for row ← 1 to 2^j do
    Reconstruction(c[row, 1 .. 2^k])
  end for
end procedure
```

The second type of two-dimensional wavelet transform, called the *nonstandard decomposition* [5], alternates between operations on rows and columns. First, we perform one step of horizontal pairwise averaging and differencing on the pixel values in each row of the image. Next, we apply vertical pairwise averaging and differencing to each column of the result. To complete the transformation, we repeat this process recursively only on the quadrant containing averages in both directions. Figure 3.2 shows all the steps involved in the nonstandard decomposition procedure below.

```
procedure NonstandardDecomposition(c: array [1 .. 2^j, 1 .. 2^j] of reals)
  c ← c/2^j    (normalize input coefficients)
  g ← 2^j
  while g ≥ 2 do
    for row ← 1 to g do
      DecompositionStep(c[row, 1 .. g])
    end for
    for col ← 1 to g do
      DecompositionStep(c[1 .. g, col])
    end for
    g ← g/2
  end while
end procedure
```

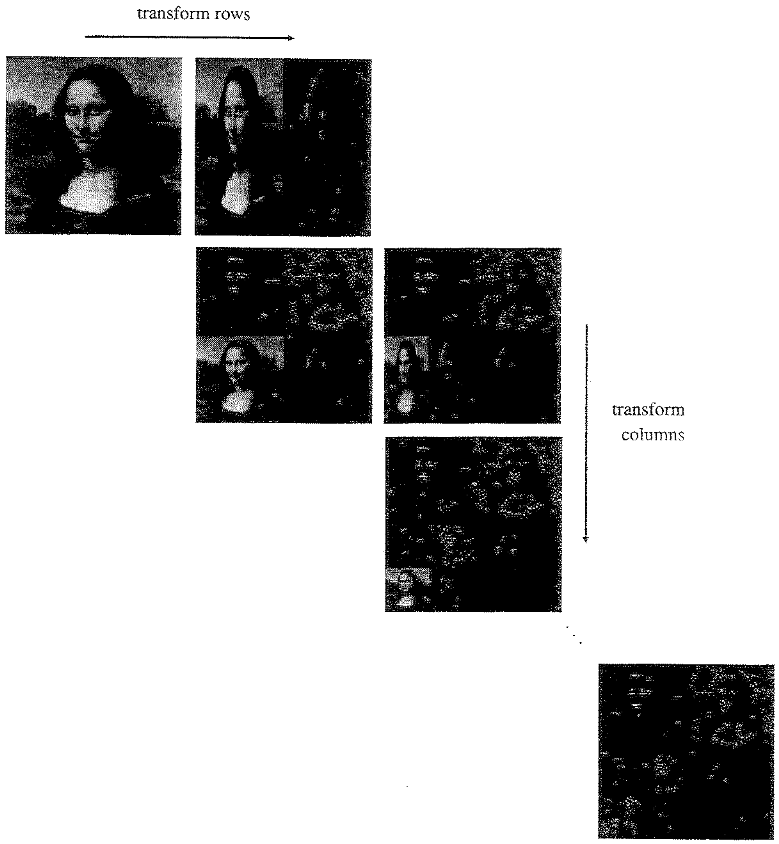


FIGURE 3.2 Nonstandard decomposition of an image.

Here is the pseudocode to perform the nonstandard reconstruction:


```
procedure NonstandardReconstruction(c: array [1 . . 2^j, 1 . . 2^j] of reals)
  g ← 2
  while g ≤ 2^j do
    for col ← 1 to g do
      ReconstructionStep(c[1 . . g, col])
    end for
    for row ← 1 to g do
      ReconstructionStep(c[row, 1 . . g])
    end for
    g ← 2g
  end while
  c ← 2^j c    (undo normalization)
end procedure
```

3.2 Two-dimensional Haar basis functions

The two methods of decomposing a two-dimensional image yield coefficients that correspond to two different sets of basis functions. The standard decomposition of an image gives coefficients for a basis formed by the *standard construction* of a two-dimensional basis. Similarly, the nonstandard decomposition gives coefficients for the *nonstandard construction* of basis functions [5].

The standard construction of a two-dimensional wavelet basis consists of all possible tensor products of one-dimensional basis functions. For example, when we start with the one-dimensional Haar basis for V^2, we get the two-dimensional basis for V^2 shown in Figure 3.3. Note that if we apply the standard construction to an orthonormal basis in one dimension, we get an orthonormal basis in two dimensions.

The nonstandard construction of a two-dimensional basis proceeds by first defining a two-dimensional scaling function,

$$\phi\phi(x, y) := \phi(x)\,\phi(y)$$

and three wavelet functions,

$$\phi\psi(x, y) := \phi(x)\,\psi(y)$$
$$\psi\phi(x, y) := \psi(x)\,\phi(y)$$
$$\psi\psi(x, y) := \psi(x)\,\psi(y)$$

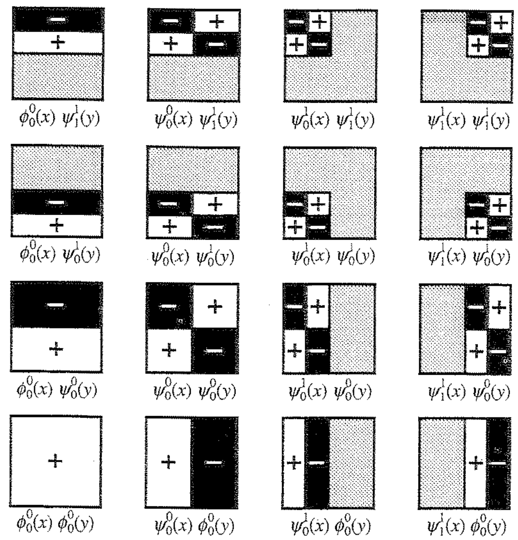


FIGURE 3.3 The standard construction of a two-dimensional Haar wavelet basis for V^2. In the unnormalized case, functions are +1 where plus signs appear, −1 where minus signs appear, and 0 in gray regions.

We now denote levels of scaling with a superscript j (as we did in the one-dimensional case) and horizontal and vertical translations with a pair of subscripts k and ℓ. The nonstandard basis consists of a single coarse scaling function $\phi\phi_{0,0}^0(x, y) := \phi\phi(x, y)$ along with scales and translates of the three wavelet functions $\phi\psi$, $\psi\phi$, and $\psi\psi$:

$$\phi\psi_{k\ell}^j(x, y) := 2^j \phi\psi(2^j x - k, 2^j y - \ell)$$
$$\psi\phi_{k\ell}^j(x, y) := 2^j \psi\phi(2^j x - k, 2^j y - \ell)$$
$$\psi\psi_{k\ell}^j(x, y) := 2^j \psi\psi(2^j x - k, 2^j y - \ell)$$

The constant 2^j normalizes the wavelets to give an orthonormal basis. The nonstandard construction results in the basis for V^2 shown in Figure 3.4.

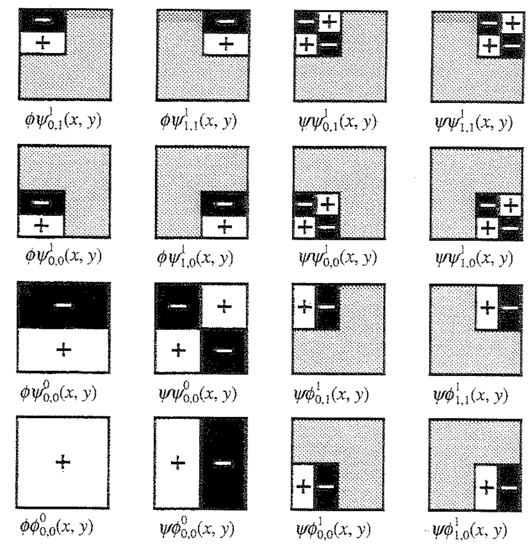


FIGURE 3.4 The nonstandard construction of a two-dimensional Haar wavelet basis for V^2.

We have presented both the standard and nonstandard approaches to wavelet transforms and basis functions because they each have advantages. The standard decomposition of an image is appealing because it can be accomplished simply by performing one-dimensional transforms on all the rows and then on all the columns. On the other hand, it is slightly more efficient to compute the nonstandard decomposition of an image. For an $m \times m$ image, the standard decomposition requires $4(m^2 - m)$ assignment operations, while the nonstandard decomposition requires only $\frac{8}{3}(m^2 - 1)$ assignment operations.

comparison between standard and non-standard

Another consideration is the support of each basis function, meaning the portion of each function's domain where that function is nonzero. All of the nonstandard Haar basis functions have square supports, while some of the standard basis functions have nonsquare supports. Depending upon the application, one of these choices may be preferable to the other.

3.3 Wavelet image compression

We defined compression in Section 2.4 as the representation of a function using fewer basis function coefficients than were originally given. The method we discussed for one-dimensional functions applies equally well to images, which we treat as the coefficients corresponding to a two-dimensional piecewise-constant basis. The approach presented here is only introductory; for a more complete treatment of wavelet image compression, see the article by DeVore et al. [27]. Once again, we note that we are dealing only with the transformation and quantization of coefficients and not with how they are encoded.

Wavelet image compression using the L^2 norm can be summarized in three steps:

1. Compute coefficients $c_1, \ldots, c_m$ representing an image in a normalized two-dimensional Haar basis.
2. Sort the coefficients in order of decreasing magnitude to produce the sequence $c_{\pi(1)}, \ldots, c_{\pi(m)}$.
3. Given an allowable L^2 error ε and starting with $\hat{m} = m$, find the smallest $\hat{m}$ for which

$$\sum_{i=\hat{m}+1}^{m} (c_{\pi(i)})^2 \le \varepsilon^2$$

The first step is accomplished by applying either of the two-dimensional Haar wavelet transforms described in Section 3.1, being sure to use normalized basis functions. Any standard sorting technique will work for the second step, and any standard search can be used for the third step. However, for large images sorting becomes exceedingly slow. The pseudocode below outlines a more efficient method of accomplishing steps 2 and 3, which uses a binary search strategy to find a threshold τ below which coefficients can be truncated.

The procedure takes as input a one-dimensional array of coefficients **c** (with each coefficient corresponding to a two-dimensional basis function) and an error tolerance ε. For each guess at a threshold τ, the algorithm computes the square of the L^2 error that would result from discarding coefficients smaller in magnitude than τ. This squared error s is compared to ε^2 at each iteration to decide whether the binary search should continue in the upper or lower half of the current interval. The algorithm halts when the current interval is so narrow that the number of coefficients to be discarded no longer changes.


```
procedure Compress(c: array [1 .. m] of reals; ε: real)
  τ_min ← min{ |c[i]| }
  τ_max ← max{ |c[i]| }
  do
    τ ← (τ_min + τ_max)/2
    s ← 0
    for i ← 1 to m do
      if |c[i]| < τ then s ← s + |c[i]|^2
    end for
    if s < ε^2 then τ_min ← τ else τ_max ← τ
  until τ_min ≈ τ_max
  for i ← 1 to m do
    if |c[i]| < τ then c[i] ← 0
  end for
end procedure
```

The binary search algorithm given above was used to produce the images in Figure 3.5. These images demonstrate the high compression ratios wavelets offer as well as some of the artifacts they introduce.

DeVore et al. [27] suggest that the L^1 norm is best suited to the task of image compression. Here is a pseudocode fragment for a "greedy" L^1 compression scheme, which works by accumulating in a two-dimensional array $\Delta[x, y]$ the error introduced by discarding a coefficient and checking whether this error has exceeded a user-specified threshold:

```
for each pixel (x, y) do
  Δ[x, y] ← 0
end for
for i ← 1 to m do
  Δ′ ← Δ + error from discarding c[i]
  if Σ_{x,y} |Δ′[x, y]| < ε then
    c[i] ← 0
    Δ ← Δ′
  end if
end for
```

(a) (b) (c) (d)

FIGURE 3.5 L^2 Haar wavelet image compression: The original image (a) can be represented using (b) 19% of its wavelet coefficients, with 5% relative L^2 error; (c) 3% of its coefficients, with 10% relative L^2 error; (d) 1% of its coefficients, with 15% relative L^2 error.

Note that this algorithm's results depend on the order in which coefficients are visited. One could imagine obtaining very different images (and degrees of compression) by varying the order—for example, by starting with the finest scale coefficients rather than the smallest coefficients. One could also imagine running a more sophisticated constrained optimization procedure to select the minimum number of coefficients subject to the error bound.

3.4 Color images

Up to now, our discussion of images has encompassed only single-component gray-scale images. However, wavelet transforms and compression techniques apply equally well to color images with three color components. For example, the pseudocode given above for L^2 compression will work for a color image if we perform a wavelet transform independently on each of the three color components of the image and treat the results as an array of vector-valued wavelet coefficients. Then, instead of using the absolute value of a scalar coefficient in the pseudocode, we use the L^2 norm (the usual vector magnitude) of a vector-valued coefficient. Examples of color images compressed using this algorithm are shown in Color Plate 1.

Furthermore, there are a number of ways in which color information can be used to obtain a wavelet transform that is even more sparse than those we have discussed. For instance, by first converting the pixel values in an image from *RGB* colors to *YIQ* colors [38], we can separate the luminance information (Y) from the chromatic information (I and Q). Once we compute the wavelet transform, we can apply an L^2 compression procedure to each of the components of the image separately. Since human perception is most sensitive to variation in Y and least sensitive to variation in Q, we can permit the compression scheme to tolerate a

larger error in the Q component of the compressed image, thereby increasing the amount of compression. (The same principle allows U.S. color television signals to be broadcast with bandwidths of 4 MHz for Y, 1.5 MHz for I, and 0.6 MHz for Q.)

3.5 Summary

In this and the previous chapter, we have described Haar wavelets in one and two dimensions, as well as how they can be used to compress functions and images. The Haar basis is also useful for image editing and querying, as described in the next two chapters, as well as for global illumination, as described in Chapter 13.

The theoretical exposition of wavelets will continue in Chapters 6 and 7, which present the theory of subdivision curves and show how this theory can be used in developing a more complete mathematical framework for multiresolution analysis.