CHAPTER 13

Discrete Image Transforms

13.1 INTRODUCTION

The discrete Fourier transform (DFT), introduced in Chapter 10, is but one of a number of discrete linear transformations that prove useful in digital image processing. In this chapter, we examine the topic more generally, developing several other transforms and some of their properties and applications as well.

Images of interest normally occur in continuous form and must be viewed that way as well. Since we are limited to working with a discrete representation of a continuous image, much of digital image processing requires that we keep sampling and interpolation considerations in mind while processing the discrete data. Some applications, however, allow us to treat the digital image as a discrete entity, without particular regard for the history of its origin or for the underlying continuous image.

One such application is image compression. Here, one wishes to encode an image into a more compact data format, either with no loss, or with only a tolerable loss of information content. Normally, considerations of optics, sampling, and interpolation, regarding the digitization and display of the image, are not of immediate concern, and the digital image can be treated merely as a data file.

A *representation* of an image is a particular embodiment of the data that defines the image. It is a presentation of the image data in a particular form or format. A digital image can be represented as a matrix or, with row stacking, as a vector.

13.2 LINEAR TRANSFORMATIONS

13.2.1 One-Dimensional Discrete Linear Transformations

Definition. If $\mathbf{x}$ is an N-by-1 vector and $\mathbf{T}$ is an N-by-N matrix, then

$$y_i = \sum_{j=0}^{N-1} t_{i,j} x_j \quad \text{or} \quad \mathbf{y} = \mathbf{T}\mathbf{x} \tag{1}$$

where $i = 0, \ldots, N-1$ defines a linear transformation of the vector $\mathbf{x}$. The matrix $\mathbf{T}$ is also called the *kernel matrix* of the transformation. Note that this use of the word *kernel* is different from its use in the term *convolution kernel* discussed in Sec. 9.3.4.

The result of the transformation is another N-by-1 vector, $\mathbf{y}$. The transformation is linear because $\mathbf{y}$ is formed by a first-order summation of the input elements. Each element y_i is the inner product of the input vector $\mathbf{x}$ with the ith row of $\mathbf{T}$.

Example. A simple example of a linear transformation is the rotation of a vector in a two-dimensional coordinate system. (See Chapter 8.) Here,

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{2}$$

rotates the vector $\mathbf{x}$ through the angle θ.

Inversion. After the transformation, the original vector can be recovered by the inverse transformation

$$\mathbf{x} = \mathbf{T}^{-1}\mathbf{y} \tag{3}$$

provided that $\mathbf{T}$ is nonsingular. As before, each element of $\mathbf{x}$ is an inner product, this time between $\mathbf{y}$ and a row of $\mathbf{T}^{-1}$. For the foregoing example, this amounts to a rotation through the same angle in the reverse direction.

13.2.1.1 Unitary Transforms

For a given vector length N, there are infinitely many transformation matrices $\mathbf{T}$ that could be used. The more useful ones, however, belong to a class having certain properties.

If $\mathbf{T}$ is a unitary matrix, then

$$\mathbf{T}^{-1} = \mathbf{T}^{*t} \quad \text{and} \quad \mathbf{T}\mathbf{T}^{*t} = \mathbf{T}\mathbf{T}^{*t}\mathbf{T} = \mathbf{I} \tag{4}$$

where $*$ indicates complex conjugation of each of the elements of $\mathbf{T}$ and the ^{t} indicates the transpose operation. If $\mathbf{T}$ is unitary and has only real elements, then it is an orthogonal matrix, and it follows that

$$\mathbf{T}^{-1} = \mathbf{T}^{t} \quad \text{and} \quad \mathbf{T}\mathbf{T}^{t} = \mathbf{T}^{t}\mathbf{T} = \mathbf{I} \tag{5}$$

Notice that the i,jth element of $\mathbf{T}\mathbf{T}^t$ is the inner product of rows i and j of $\mathbf{T}$. Eq. (5) implies that this is zero unless $i = j$, in which case it is unity. Thus, the rows of $\mathbf{T}$ are a set of orthonormal vectors.

Example: The one-dimensional DFT. The one-dimensional DFT is an example of a unitary transform, since

$$\mathbf{F}_k = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} f_i \exp\left(-j2\pi k \frac{i}{N}\right) \quad \text{or} \quad \mathbf{F} = \mathfrak{W}\mathbf{f} \tag{6}$$

where $\mathfrak{W}$ is a unitary (but not orthogonal) matrix with (complex) elements

$$\omega_{i,k} = \frac{1}{\sqrt{N}} \exp\left(-j2\pi k \frac{i}{N}\right) \tag{7}$$

Interpretation. Normally, the transform matrix **T** is non-singular (i.e., rank(**T**) = N), so as to make the transform invertible, as per Eq. (3). Then the rows of **T** form an orthonormal basis (a set of orthonormal basis vectors, or unit vectors) for the N-dimensional vector space of all N-by-1 vectors. This means that any N-by-1 sequence can be viewed as representing a vector from the origin to a point in N-dimensional space. Furthermore, any transform of the form of Eq. (1) can be viewed as a coordinate transformation, rotating the vector in N-space without changing its length.

In summary, then, a unitary linear transformation generates **y**, a vector of N *transform coefficients*, each of which is computed as the inner product of the input vector **x** with one of the rows of the transform matrix **T**. The inverse transform is computed similarly, as a set of inner products of the transform coefficient vector with the rows of the inverse transform matrix.

The forward transformation is generally considered to be a process of *analysis,* breaking the signal vector down into its elemental components. These fundamental components are naturally in the form of the basis vectors. The transform coefficients specify how much of each component is found to be present in the mixture that comprises the particular vector being decomposed.

The inverse transformation, on the other hand, is often considered a process of *synthesis,* reassembling the original vector from its components via summation. Here, the transform coefficients specify the proper amount of each basis vector that must be added to the mixture so as to reconstruct the input vector accurately and completely.

A key to this process is the principle that any vector can be uniquely decomposed into a set of normal basis vectors of the proper amplitude and later reconstituted by adding these components back together to reconstruct the original. It is significant that the number of transform coefficients is equal to the number of elements in the vector. Thus, the number of degrees of freedom is the same before and after the transformation, and information is neither created nor destroyed by the process.

A transformed vector is a representation of the original vector. Since it contains the same number of elements (and thus has the same number of degrees of freedom) as the original, and since the original can be recovered from it without error, it can be considered an alternative form of expressing the original vector. This chapter considers several alternative ways of representing digital signals and images, and the usefulness of each.

13.2.2 Two-Dimensional Discrete Linear Transformations

In two dimensions, the general linear transformation that takes the N-by-N matrix **F** into the transformed matrix **G** (also N by N) is

$$G_{m,n} = \sum_{i=0}^{N-1} \sum_{k=0}^{N-1} F_{i,k} \mathfrak{I}(i, k, m, n) \tag{8}$$

where i, k, m, and n are discrete variables that range from 0 to $N-1$ and $\mathfrak{I}(i, k, m, n)$ is the kernel function of the transformation.

$\mathfrak{I}(i, k, m, n)$ can be thought of as an N^2-by-N^2 block matrix having N rows of N blocks, each of which is an N-by-N matrix. The blocks are indexed by m, n and the elements of each N-by-N sub-matrix by i, k (See Figure 13–1.)

If $\mathfrak{I}(i, k, m, n)$ can be separated into the product of rowwise and columnwise component functions—that is, if

$$\mathfrak{I}(i, k, m, n) = T_r(i, m)T_c(k, n) \tag{9}$$

then the transformation is called *separable.* This means that it can be carried out in two steps—a rowwise operation followed by a columnwise operation (or vice versa):

$$G_{m,n} = \sum_{i=0}^{N-1} \left[\sum_{k=0}^{N-1} F_{i,k} T_c(k, n) \right] T_r(i, m) \tag{10}$$

Further, if the two component functions are identical, the transform is also called *symmetric* (not to be confused with a symmetric matrix). Then

$$\mathfrak{I}(i, k, m, n) = T(i, m)T(k, n) \tag{11}$$

and Eq. (8) can be written as

$$G_{m,n} = \sum_{i=0}^{N-1} T(i, m) \left[\sum_{k=0}^{N-1} F_{i,k} T(k, n) \right] \quad \text{or} \quad \mathbf{G} = \mathbf{TFT} \tag{12}$$

where $\mathbf{T}$ is a unitary matrix, called the *kernel matrix* of the transform, as before. We use this notation throughout the chapter, to signify a general, separable, symmetric unitary transform.

The inverse transform is

$$\mathbf{F} = \mathbf{T}^{-1}\mathbf{G}\mathbf{T}^{-1} = \mathbf{T}^{*t}\mathbf{G}\mathbf{T}^{*t} \tag{13}$$

and it recovers $\mathbf{F}$ exactly.

Example: The Two-dimensional DFT. The two-dimensional DFT is a separable and symmetric unitary transform. In this case, $\mathbf{T}$ in Eq. (12) becomes the matrix $\mathcal{W}$ from Eq. (7).

The inverse DFT uses $\mathcal{W}^{-1}$, which is simply the conjugate transpose of $\mathcal{W}$. The discrete Fourier transform pair is thus expressed as

$$\mathbf{G} = \mathcal{W}\mathbf{F}\mathcal{W} \quad \text{and} \quad \mathbf{F} = \mathcal{W}^{*t}\mathbf{G}\mathcal{W}^{*t} \tag{14}$$

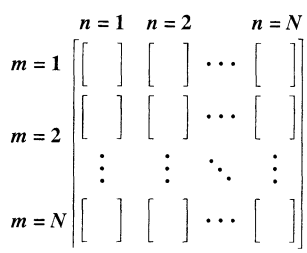

Figure 13–1 The kernel matrix

13.2.2.1 Orthogonal Transformations

Unlike the Fourier transform, many transforms have only real elements in their kernel matrix **T**. A unitary matrix with real elements is orthogonal, and the inverse transformation becomes simply

$$\mathbf{F} = \mathbf{T}^t\mathbf{G}\mathbf{T}^t \tag{15}$$

If **T** is a symmetric matrix, as is often the case, then the forward and inverse transforms are identical, so that

$$\mathbf{G} = \mathbf{TFT} \quad \text{and} \quad \mathbf{F} = \mathbf{TGT} \tag{16}$$

13.3 BASIS FUNCTIONS AND BASIS IMAGES

The primary difference between any two unitary transforms is the choice of basis functions, that is, the rows of **T**. Here, we examine basis functions in more detail.

13.3.1 Basis Functions

The rows of the kernel matrix form a set of basis vectors for an N-dimensional vector space. The rows are orthonormal; that is,

$$\mathbf{T}\mathbf{T}^{*t} = \mathbf{I} \quad \text{or} \quad \sum_{i=0}^{N-1} T_{j,i}T^*_{k,i} = \delta_{j,k} \tag{17}$$

where $\delta_{j,k}$ is the Kronecker delta.

While any set of orthonormal vectors will serve for a linear transform, normally the entire set is derived from the same basic functional form. The Fourier transform, for example, uses the complex exponential as its prototypical basis function. The individual basis functions differ only in frequency.

Any vector in the space can be expressed as a weighted sum of unit-length basis vectors. Any one-dimensional (N-by-1) unitary transform, then, corresponds to a rotation of a vector in an N-dimensional vector space. Further, since an N-by-N image matrix can be row-stacked to form an N^2-by-1 vector, any two-dimensional, symmetric, separable unitary transform corresponds to a rotation of a vector in an N^2-dimensional vector space.

13.3.2 Basis Images

The inverse two-dimensional transform can be viewed as reconstructing the image by summing a set of properly weighted *basis images*. Each element in the transform matrix, **G**, is the coefficient by which the corresponding basis image is multiplied in the summation.

A basis image can be generated by inverse transforming a coefficient matrix containing only one nonzero element, which is set to unity. There are N^2 such matrices, and these produce N^2 basis images. Let one such coefficient matrix be

$$\mathbf{G}^{p,q} = \{\delta_{i-p,j-q}\} \tag{18}$$

where i and j are the row and column indices and p and q are integers that specify the location of the nonzero element. Then the inverse transform [Eq. (13)] is

$$F_{m,n} = \sum_{i=0}^{N-1} T(i,m) \left[\sum_{k=0}^{N-1} \delta_{i-p,k-q} T(k,n) \right] = T(p,m)T(q,n) \tag{19}$$

Thus, for a separable unitary transform, each basis image is the outer product of two rows of the transform matrix.

As with one-dimensional signals, the basis images can be thought of as a set of basic components into which any image can be decomposed. They are also the building blocks from which any image can be reassembled. The forward transform does the decomposition by determining the coefficients. The inverse transform does the reconstitution by summing the basis images, weighted by those coefficients.

Since infinitely many sets of basis images exist, infinitely many transforms exist as well. Thus, a particular set of basis images takes on profound importance only in the context of a particular transform.

13.4 SINUSOIDAL TRANSFORMS

For reasons mentioned in Chapter 10, the Fourier transform has emerged as the single most important transform in digital imaging. It has, however, several relatives that also use sinusoidal basis functions. These are introduced in this section, after a brief review of the discrete Fourier transform.

13.4.1 The Discrete Fourier Transform

Introduced in Chapter 10, the DFT is considered again here, in the context of separable unitary transforms, to enable us to draw comparisons between it and other transforms of the same type.

The kernel matrix for the DFT [recall Eqs. (6) and (7)] is

$$\mathcal{W} = \begin{bmatrix} w_{0,0} & \cdots & w_{0,N-1} \\ \vdots & \ddots & \vdots \\ w_{N-1,0} & \cdots & w_{N-1,N-1} \end{bmatrix} \tag{20}$$

where

$$w_{i,k} = \frac{1}{\sqrt{N}} e^{-j2\pi \frac{ik}{N}} \tag{21}$$

Because of the periodic nature of the imaginary exponential, $\mathcal{W}$ is unitary.

In one dimension, the forward and inverse DFTs are

$$\mathbf{F} = \mathcal{W}\mathbf{f} \quad \text{and} \quad \mathbf{f} = \mathcal{W}^{*t}\mathbf{F} \tag{22}$$

where $\mathbf{f}$ and $\mathbf{F}$ are N-by-1 signal and spectrum vectors, respectively. If $\mathbf{f}$ is real, $\mathbf{F}$ will, in general, have complex elements. Only if $\mathbf{f}$ has the proper symmetry (discussed next) will $\mathbf{F}$ be real.

13.4.1.1 The Spectrum Vector

Figure 13–2 shows where the different frequency components occur in the spectrum vector **F**, when **f** is real. The zero-frequency component and the highest frequency component (corresponding to the Nyquist frequency) appear only once. The remaining components are duplicated as complex conjugates. (Recall that the spectrum of a real function is a Hermite function.) If $\mathbf{F}^t$ is viewed as a row vector, the first $N/2 + 1$ elements are the right half of the spectrum, while the last $N/2 - 1$ elements are the left half. The frequency corresponding to the ith element of **F** is

$$s_i = \begin{cases} \dfrac{2i}{N} f_N & 0 \le i \le N/2 \\ \dfrac{2(N-i)}{N} f_N & N/2 + 1 \le i \le N - 1 \end{cases} \tag{23}$$

where f_N is the Nyquist (folding) frequency (half the sampling frequency). If the last $N/2 - 1$ elements of **f** form a mirror image of elements 1 through $N/2 - 1$, then **F** is even, and **F** will be real-valued.

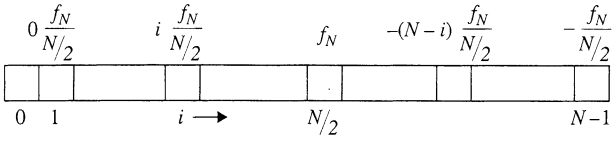

Figure 13–2 Location of the different frequency components in the spectrum vector

One can rotate the elements of **F** by the amount $N/2$, using a circular right (or left) shift operation, to produce a vector suitable for plotting the spectrum. In that case, the zero-frequency element is located at $N/2$, and frequency increases in both directions from there. The Nyquist frequency element appears only at F_0.

The shift theorem of the Fourier transform (Sec. 10.2.3) provides another way of achieving the same result. Applying the theorem to a shift in the frequency domain yields

$$\begin{aligned} F(u) \Leftrightarrow f(x) \Rightarrow F(u - u_0) &\Leftrightarrow \exp\left(j2\pi x \frac{u_0}{N}\right) f(x) \\ &= \exp(j\pi x) f(x) = (-1)^x f(x) \end{aligned} \tag{24}$$

where the amount of shift is $u_0 = N/2$. This means that we have merely to change the signs of the odd-numbered elements of $f(x)$ prior to executing the DFT. Doing so leaves the spectrum properly shifted for plotting.

13.4.1.2 The Two-Dimensional DFT

In two dimensions, the forward and inverse DFTs are

$$\mathbf{G} = \mathcal{W}\mathbf{F}\mathcal{W} \quad \text{and} \quad \mathbf{F} = \mathcal{W}^{*t}\mathbf{G}\mathcal{W}^{*t} \tag{25}$$

where **F** is an image in matrix form and **G** is its spectrum matrix.

Figure 13–3 shows where the various spatial frequency terms are located in the spectrum matrix **G**. Rearrangement of the four quadrants, as shown in the figure, makes displaying the spectrum more convenient. That way, zero frequency falls at the center of

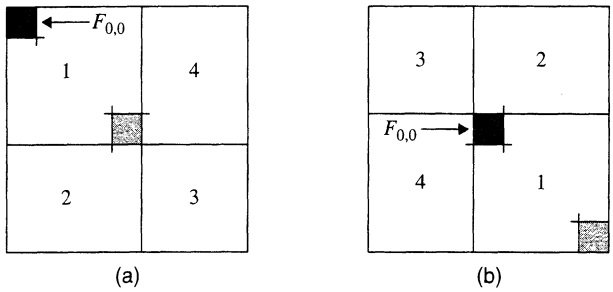

Figure 13–3 Location of the various spatial frequency terms in the spectrum matrix: (a) after transformation; (b) after rearrangement

the matrix, and frequency increases radially from there. In two dimensions, Eq. (24) generalizes to

$$F(u, v) \Leftrightarrow f(x, y) \Rightarrow F(u - N/2, v - N/2) \Leftrightarrow (-1)^{x+y} f(x, y) \tag{26}$$

and again, changing the sign of half the elements of the image matrix **F** effects the desired shift. If **F** has the symmetry shown in Figure 13–3(a), then **G** will be real valued.

13.4.2 The Discrete Cosine Transform

The discrete cosine transform (DCT) is defined in two dimensions as

$$G_c(m, n) = \alpha(m)\alpha(n) \sum_{i=0}^{N-1} \sum_{k=0}^{N-1} g(i, k) \cos\left[\frac{\pi(2i+1)m}{2N}\right] \cos\left[\frac{\pi(2k+1)n}{2N}\right] \tag{27}$$

and its inverse by

$$g(i, k) = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} \alpha(m)\alpha(n) G_c(m, n) \cos\left[\frac{\pi(2i+1)m}{2N}\right] \cos\left[\frac{\pi(2k+1)n}{2N}\right] \tag{28}$$

where the coefficients are

$$\alpha(0) = \sqrt{\frac{1}{N}} \quad \text{and} \quad \alpha(m) = \sqrt{\frac{2}{N}} \quad \text{for} \quad 1 \le m \le N \tag{29}$$

Like the DFT, the DCT can be expressed as a unitary matrix operation in the form

$$\mathbf{G_c} = \mathbf{CgC} \tag{30}$$

where the kernel matrix has elements

$$C_{i,m} = \alpha(m) \cos\left[\frac{\pi(2i+1)m}{2N}\right] \tag{31}$$

Also like the DFT, the DCT can be computed by a fast algorithm [1–3]. Unlike the DFT, the DCT is real valued. It has found wide usage in image compression, for reasons pointed out later in the chapter.

13.4.3 The Sine Transform

Jain [4] introduced the discrete sine transform (DST), defined as

$$G_s(m,n) = \frac{2}{N+1}\sum_{i=0}^{N-1}\sum_{k=0}^{N-1} g(i,k)\sin\left[\frac{\pi(i+1)(m+1)}{N+1}\right]\sin\left[\frac{\pi(k+1)(n+1)}{N+1}\right] \quad (32)$$

and

$$g(i,k) = \frac{2}{N+1}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} G_s(m,n)\sin\left[\frac{\pi(i+1)(m+1)}{N+1}\right]\sin\left[\frac{\pi(k+1)(n+1)}{N+1}\right] \quad (33)$$

The DST has unitary kernel matrix elements

$$T_{i,k} = \sqrt{\frac{2}{N+1}}\sin\left[\frac{\pi(i+1)(k+1)}{N+1}\right] \quad (34)$$

Unlike the other sinusoidal transforms, the DST is most conveniently computed for $N = 2^p - 1$, where p is an integer. Then it can be taken as the imaginary part of a specially constructed $(2N + 2)$-point FFT [5].

The DST has a fast implementation algorithm [6] and properties that prove useful in certain image compression problems, as discussed later in the chapter.

13.4.4 The Hartley Transform

In 1942, Hartley introduced a continuous integral transform as an alternative to the Fourier transform [7]. Bracewell later defined an analogous discrete unitary transform based on the Hartley transform [8]. The forward two-dimensional discrete Hartley transform (DHT)

$$G_{m,n} = \frac{1}{N}\sum_{i=0}^{N-1}\sum_{k=0}^{N-1} g_{i,k}\,\mathrm{cas}\left[\frac{2\pi}{N}(im+kn)\right] \quad (35)$$

and the inverse two-dimensional DHT

$$g_{i,k} = \frac{1}{N}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} G_{m,n}\,\mathrm{cas}\left[\frac{2\pi}{N}(im+kn)\right] \quad (36)$$

are identical and use the basis function

$$\mathrm{cas}(\theta) = \cos(\theta) + \sin(\theta) = \sqrt{2}\cos(\theta - \pi/4) \quad (37)$$

which is a cosine shifted 45 degrees to the right.

The kernel matrix of the Hartley transform has elements

$$T_{i,k} = \frac{1}{\sqrt{N}}\left[\mathrm{cas}\left(2\pi\frac{ik}{N}\right)\right] \quad (38)$$

Whereas the DFT transforms N real numbers into N complex numbers with conjugate symmetry, the discrete Hartley transform produces N real numbers.

As one might expect, the DHT is closely related to the DFT. In Chapter 10, we saw that the Hartley transform is simply the real part minus the imaginary part of the corresponding Fourier transform. Likewise, the Fourier transform is the even part minus j times the odd part of the Hartley transform.

The convolution theorem of the Hartley transform is only slightly more complicated than that of the Fourier transform. It is expressed as

$$g(x) = f(x) * h(x) \Leftrightarrow G(v) = F(v)H_e(v) + F(-v)H_o(v) \tag{39}$$

where $F(v)$ and $G(v)$ are the Hartley transforms of $f(x)$ and $g(x)$, respectively, and $H_e(v)$ and $H_o(v)$ are the even and odd components, respectively, of the Hartley transform of $h(x)$. (See Sec. 10.2.1 for a definition of even and odd components.)

In the common case where one of the functions is even, the second term of Eq. (39) drops out, and convolution corresponds to multiplication in the Hartley transform domain, just as it does with the Fourier transform in the frequency domain.

The DHT is a computational alternative to the DFT. There is a fast algorithm for the Hartley transform [9]. For linear filtering applications—particularly if the kernel is symmetric—the DHT can significantly reduce the computational work load, since it avoids complex arithmetic.

13.4.5 Other Sinusoidal Transforms

Jain [10] has introduced a family of unitary transforms having sinusoidal basis functions. The DFT, the DCT, and the DST belong to this family.

13.5 RECTANGULAR WAVE TRANSFORMS

Several transforms of interest in discrete image processing use basis functions that are variations of the square wave rather than sinusoids. In general, these are fast to compute, since many of the multiplication operations become trivial.

In this section, we introduce the Hadamard, Walsh, slant, and Haar transforms. The Haar transform differs fundamentally from the other three, and it is discussed further, in the context of wavelet transforms, in the next chapter.

13.5.1 The Hadamard Transform

The Hadamard transform [11–15] is a symmetric, separable unitary transformation that has only +1 and −1 as elements in its kernel matrix. It exists for $N = 2^n$, where n is an integer.

For the two-by-two case, the kernel matrix is

$$\frac{1}{\sqrt{2}}\mathbf{H}_2 = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{40}$$

and for successively larger N, these can be generated from the block matrix form

$$\frac{1}{\sqrt{N}}\mathbf{H}_N = \frac{1}{\sqrt{N}}\begin{bmatrix} \mathbf{H}_{N/2} & \mathbf{H}_{N/2} \\ \mathbf{H}_{N/2} & -\mathbf{H}_{N/2} \end{bmatrix} \tag{41}$$

For any size $N = 2^n$, the matrix contains only elements that are ±1, provided that the $N^{-1/2}$ factor is kept out in front. This makes the transform less expensive to compute.

For $N = 8$, for example, the Hadamard kernel matrix is

$$H_8 = \frac{1}{2\sqrt{2}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \begin{matrix} 0 \\ 7 \\ 3 \\ 4 \\ 1 \\ 6 \\ 2 \\ 5 \end{matrix} \tag{42}$$

where the column to the right shows the number of sign changes along the corresponding row. Notice that these are different for each row. This sign change count is called the *sequency* of the row [16].

We can reorder the rows to make sequency increase uniformly with row number, much as frequency increases with the Fourier kernel. This yields a transform that is somewhat easier to interpret. The kernel of the *ordered Hadamard transform,* for $N = 8$, is thus

$$H_8 = \frac{1}{2\sqrt{2}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \end{bmatrix} \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{matrix} \tag{43}$$

13.5.2 The Walsh Transform

The Hadamard transform basis functions are actually Walsh functions [17]. Thus, the Hadamard transform is also referred to as the Walsh transform.

13.5.3 The Slant Transform

The slant transform [18] was designed to have not only a constant first basis function, but a linear second basis function as well (Figure 13–4). The slanted second basis function matches the linearly sloping background that is present in many images.

The unitary kernel matrix for the slant transform is obtained by starting with the two-by-two Haar or Hadamard matrix

$$S_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{44}$$

and iterating it according to the schema

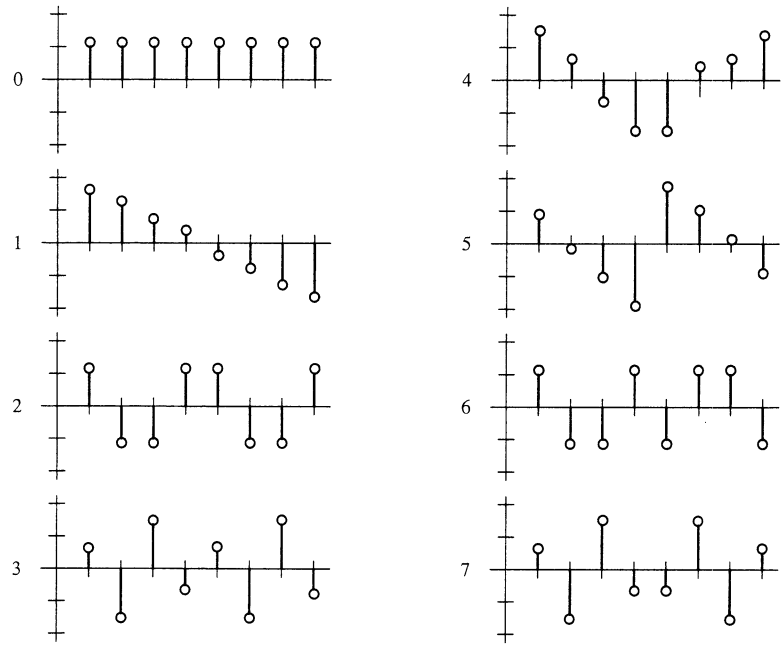

Figure 13–4 The slant transform basis functions for $N = 8$

$$S_N = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & \mathbf{0} & 1 & 0 & \mathbf{0} \\ a_N & b_N & \mathbf{0} & -a_N & b_N & \mathbf{0} \\ \mathbf{0} & & \mathbf{I} & \mathbf{0} & & \mathbf{I} \\ 0 & 1 & \mathbf{0} & 0 & -1 & \mathbf{0} \\ -b_N & a_N & \mathbf{0} & b_N & a_N & \mathbf{0} \\ \mathbf{0} & & \mathbf{I} & \mathbf{0} & & -\mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{S}_{N/2} & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_{N/2} \end{bmatrix} \tag{45}$$

where $\mathbf{I}$ is the identity matrix of order $N/2 - 2$ and

$$a_{2N} = \sqrt{\frac{3N^2}{4N^2 - 1}} \quad \text{and} \quad b_{2N} = \sqrt{\frac{N^2 - 1}{4N^2 - 1}} \tag{46}$$

The slant transform basis functions occur in all sequences from 0 through $N - 1$. The slant transform also has a fast transform algorithm and has been used for image compression [18].

13.5.4 The Haar Transform

The Haar transform is a symmetric, separable unitary transformation that uses Haar functions for its basis [19–21]. It exists for $N = 2^n$, where n is an integer.

Whereas the Fourier transform basis functions differ only in frequency, the Haar functions vary in both scale (width) and position. This gives the Haar transform a dual scale-position nature that is evident in its basis functions (Figure 13–5). Such a feature

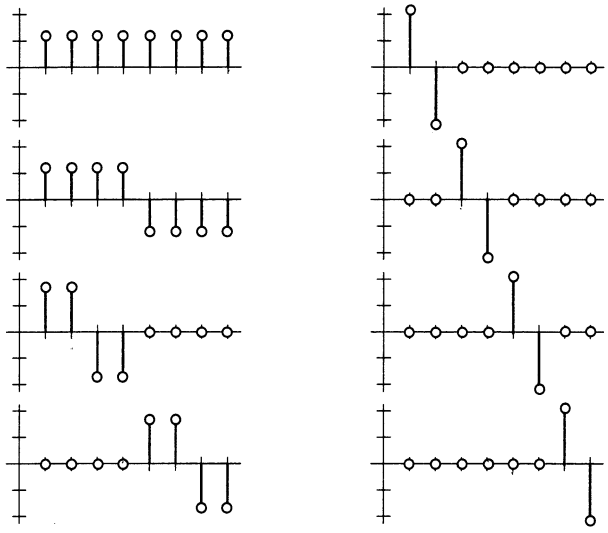

Figure 13–5 The Haar transform basis functions for $N = 8$

distinguishes it from the other transforms mentioned so far and establishes a starting point for wavelet transforms, which are introduced in the next chapter.

Basis Function Indexing. Since the Haar functions vary in two aspects (scale and position), they must be specified by a dual indexing scheme. The Haar functions are defined on the interval [0, 1] as follows. Let the integer $0 \le k \le N-1$ be specified (uniquely) by two other integers, p and q, as

$$k = 2^p + q - 1 \tag{47}$$

Notice that, under this construction, not only is k a function of p and q, but p and q are functions of k as well. For any value of $k > 0$, 2^p is the largest power of 2 such that $2^p \le k$, and $q - 1$ is the remainder.

The Haar functions are defined by

$$h_0(x) = \frac{1}{\sqrt{N}} \tag{48}$$

and

$$h_k(x) = \frac{1}{\sqrt{N}} \begin{cases} 2^{p/2} & \dfrac{q-1}{2^p} \le x < \dfrac{q-\frac{1}{2}}{2^p} \\ -2^{p/2} & \dfrac{q-\frac{1}{2}}{2^p} \le x < \dfrac{q}{2^p} \\ 0 & \text{otherwise} \end{cases} \tag{49}$$

If we let $x = i/N$ for $i = 0, 1, \ldots, N-1$, this gives rise to a set of basis functions, each of which is an odd rectangular pulse pair, except for $k = 0$, which, as in the case of many of the other transforms discussed here, is constant. Further, the basis functions vary in both scale

(width) and position (Figure 13–5). The index p specifies the scale, while q determines the shift.

Whereas the transforms discussed so far use full-width basis functions, the Haar functions are all scaled, shifted versions of a single "prototype" function, the odd rectangular pulse pair. There are two major ramifications of this property.

First, although the basis functions are identified by the single index k, they have a dual scale-position nature that is specified by the indices p and q. Thus, it is less enlightening to plot the transform coefficients along the k-axis than it would be, for example, to plot a conventional frequency spectrum obtained with the Fourier transform.

Second, suppose a particular feature, such as an edge, is embedded in the signal at some position along the x-axis. The Fourier transform, for example, encodes this position into the phase spectrum in accordance with the shift theorem (Sec. 10.2.3). While the feature position is uniquely specified and can be recovered exactly via the inverse Fourier transform, it may not be particularly visible in any convenient display of the spectrum. (*Note:* If a single feature dominates the signal, then the phase plot will be linear, with slope related to feature position (as per the shift theorem), and this can be used to locate the feature. A multiplicity of features or the presence of noise, however, normally makes the phase plot so complicated as to be uninterpretable.)

By contrast, the Haar transform addresses lines and edges more directly, since its basis functions resemble these features. Recall that a signal, or a component thereof, which approximately matches one of the basis functions will produce, in the transform, a large coefficient corresponding to that basis function. Since the basis functions are orthonormal, that signal will produce small coefficients elsewhere. Thus, the Haar transform can call attention to specific line and edge features by their size and location.

The eight-by-eight unitary kernel matrix for the Haar transform is

$$\mathbf{Hr} = \frac{1}{\sqrt{8}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} \\ 2 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & -2 \end{bmatrix} \tag{50}$$

and the same pattern holds for larger N. Because of the many constant and zero entries in the matrix, the Haar transform is very fast to compute.

The basis images for $N = 8$ appear in Figure 13–6. Notice that the lower right quadrant searches for small features at all different locations in the image.

13.6 EIGENVECTOR-BASED TRANSFORMS

Two important transforms use basis functions that are derived from eigenanalysis. (For a more complete summary of eigenanalysis, with numerical examples, see Appendix 3.)

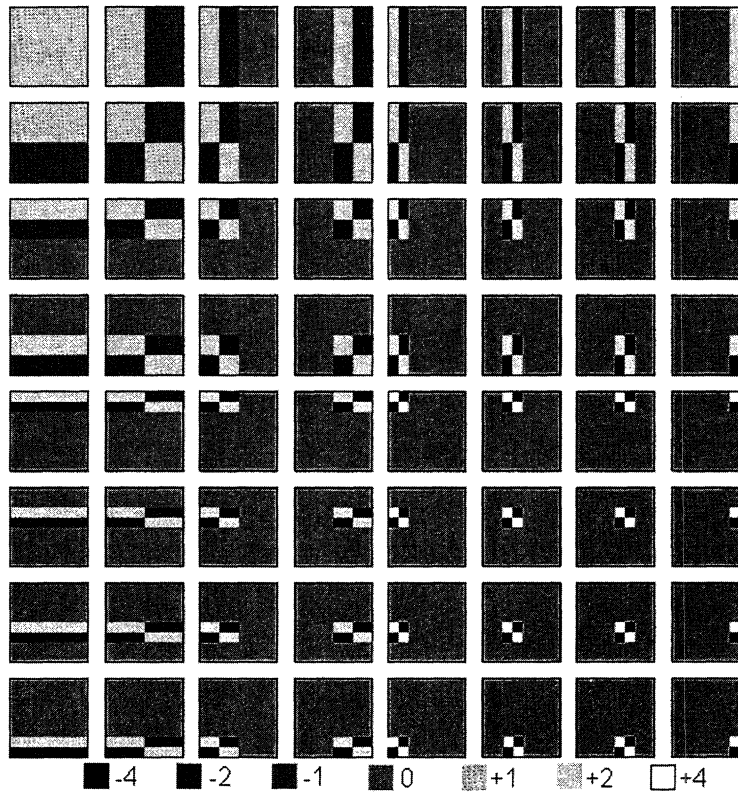

Figure 13–6 The Haar transform basis images for $N = 8$

13.6.1 Eigenanalysis

Recall that for an N-by-N matrix $\mathbf{A}$, there are N scalars, λ_k, $k = 0, \ldots, N-1$, such that

$$|\mathbf{A} - \lambda_k \mathbf{I}| = 0 \tag{51}$$

The λ_k's are called the *eigenvalues* of the matrix. (See Appendix 3.) Further, the set of N vectors $\mathbf{v}_k$ such that

$$\mathbf{A}\mathbf{v}_k = \lambda_k \mathbf{v}_k \tag{52}$$

are called the *eigenvectors* of $\mathbf{A}$. They are N by 1, and each corresponds to one of the eigenvalues. The eigenvectors form an orthonormal basis set.

13.6.2 Principal-Component Analysis

Hotelling developed a linear transformation that removes the correlation among the elements of a random vector and called it "the method of principal components" [22]. Later, Karhunen [23] and Loéve [24] developed an analogous transformation for continuous signals. This approach leads to, among other things, a discrete image transform.

Suppose $\mathbf{x}$ is an N-by-1 random vector; that is, each element x_i of $\mathbf{x}$ is a random variable. The mean vector of $\mathbf{x}$ can be estimated from a sample of L such vectors by

$$\mathbf{m}_{\mathbf{x}} \approx \frac{1}{L}\sum_{l=1}^{L} \mathbf{x}_l \tag{53}$$

and its covariance matrix by

$$\mathbf{C}_{\mathbf{x}} = \mathcal{E}\{(\mathbf{x}-\mathbf{m}_{\mathbf{x}})(\mathbf{x}-\mathbf{m}_{\mathbf{x}})^t\} \approx \frac{1}{L}\sum_{l=1}^{L} \mathbf{x}_l\mathbf{x}_l^t - \mathbf{m}_{\mathbf{x}}\mathbf{m}_{\mathbf{x}}^t \tag{54}$$

The covariance matrix is N by N, real, and symmetric. The diagonal elements are the variances of the individual random variables, while the off-diagonal elements are their covariances.

Now let the matrix $\mathbf{A}$ define a linear transformation that generates a new vector $\mathbf{y}$ from any vector $\mathbf{x}$ by

$$\mathbf{y} = \mathbf{A}(\mathbf{x}-\mathbf{m}_{\mathbf{x}}) \tag{55}$$

where $\mathbf{A}$ is constructed so that its rows are the eigenvectors of $\mathbf{C}_{\mathbf{x}}$. For convenience, we arrange the rows in order of decreasing magnitude of the corresponding eigenvalues.

The transformed vector, $\mathbf{y}$, is a random vector with zero mean. Its covariance matrix is related to that of $\mathbf{x}$ by

$$\mathbf{C}_{\mathbf{y}} = \mathbf{A}\mathbf{C}_{\mathbf{x}}\mathbf{A}^t \tag{56}$$

Since the rows of $\mathbf{A}$ are eigenvectors of $\mathbf{C}_{\mathbf{x}}$, $\mathbf{C}_{\mathbf{y}}$ is a diagonal matrix having the eigenvalues of $\mathbf{C}_{\mathbf{x}}$ along its diagonal. (This is a result of Eq. (52).) Thus,

$$\mathbf{C}_{\mathbf{y}} = \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_N \end{bmatrix} \tag{57}$$

and the λ_k are the eigenvalues of $\mathbf{C}_{\mathbf{y}}$ as well.

Because the off-diagonal elements of $\mathbf{C}_{\mathbf{y}}$ are zero, the elements of $\mathbf{y}$ are uncorrelated. Thus, the linear transformation $\mathbf{A}$ removes the correlation among the variables. Furthermore, each λ_k is the variance of y_k, the kth transformed variable.

Notice that the transform of Eq. (55) is invertible; that is, we can reconstruct a vector $\mathbf{x}$ from its transformed vector $\mathbf{y}$ by

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{y} + \mathbf{m} = \mathbf{A}^t\mathbf{y} + \mathbf{m} \tag{58}$$

The latter equality holds because $\mathbf{A}$ is unitary and real, and thus orthogonal.

13.6.2.1 Dimension Reduction

We can reduce the dimensionality of the $\mathbf{y}$ vectors by ignoring one or more of the eigenvectors that have small eigenvalues. Let $\mathbf{B}$ be the M-by-N matrix ($M < N$) formed by discarding the lower $N - M$ rows of $\mathbf{A}$, and assume, for simplicity, that $\mathbf{m} = \mathbf{0}$. Then the transformed vectors are smaller (i.e., M by 1) and are given by

$$\hat{\mathbf{y}} = \mathbf{B}\mathbf{x} \tag{59}$$

but the $\mathbf{x}$ vectors can still be reconstructed (approximately) by

$$\hat{\mathbf{x}} = \mathbf{B}^t\hat{\mathbf{y}} \tag{60}$$

The mean square error of this approximation is

$$MSE = \sum_{k=M+1}^{N} \lambda_k \tag{61}$$

that is, simply the sum of the eigenvalues corresponding to the discarded eigenvectors. Normally the eigenvalues vary considerably in magnitude, and the smaller ones can be ignored without the introduction of significant error.

13.6.3 The Karhunen-Loéve Transform

Eq. (55) defines a (one-dimensional) discrete transform. It is variously called the Karhunen-Loéve (or K-L) transform, the Hotelling transform, the eigenvector transform, or the method of principal components. We adhere to the common practice in the literature of calling it the K-L transform.

The dimension-reducing capability of the K-L transform makes it quite useful for image compression. Multispectral images, for example, have many gray-level values at every pixel, each gray-level corresponding to a different spectral band. Thus, a 1,000-by-1,000 24-channel multispectral image can be viewed as a set of one million 24-element random vectors (i.e., the pixels).

The K-L dimension-reducing technique can be applied to this set of vectors. Since the correlation between the different spectral bands of a multispectral image is commonly rather high, many of the 24 eigenvalues will be small. This means that the stack of 24 monochrome images can be represented with small error by only a few principal component images. Each of these is computed as a weighted sum of the original 24. Further, each image in the original set can be reconstructed, approximately, as a linear combination of the few principal-component images. This greatly simplifies the storage and distribution of, for example, images taken from Earth satellites.

In general, the basis images of the two-dimensional K-L transform depend upon the statistics of the particular image being transformed and cannot be written explicitly. If the image is a first-order Markov process, however, where the correlation between pixels decreases linearly with their separation distance, then the basis images for the K-L transform can be written explicitly [5,25]. The Markov assumption often fits commonly encountered images quite well. Further, as the correlation between adjacent pixels approaches unity, the K-L basis functions approach those of the discrete cosine transform [1,26]. Thus, the DCT, which is easily computed, approximates the K-L transform for commonly encountered images.

13.6.4 The SVD Transform

Any N-by-N matrix $\mathbf{A}$ can be expressed as

$$\mathbf{A} = \mathbf{U}\Lambda\mathbf{V}^t \tag{62}$$

where the columns of $\mathbf{U}$ and $\mathbf{V}$ are the eigenvectors of $\mathbf{A}\mathbf{A}^t$, and $\mathbf{A}^t\mathbf{A}$ respectively, and Λ is

an N-by-N diagonal matrix containing the *singular values* of **A** along its diagonal. (See Appendix 3 for more complete coverage of the topic.) Since **U** and **V** are orthogonal,

$$\Lambda = \mathbf{U}^t\mathbf{A}\mathbf{V} \tag{63}$$

Eq. (63) is thus the forward, and Eq. (62) the inverse, of a unitary transform pair. This transform is called *singular value decomposition* (SVD) transform [27]. If **A** is symmetric, **U** = **V**.

Notice that, unlike the transforms discussed in earlier sections, the kernel matrices **U** and **V** depend on the image **A** being transformed. In general, one must compute the eigenvectors of $\mathbf{A}\mathbf{A}^t$ and $\mathbf{A}^t\mathbf{A}$ for each image undergoing the transformation.

Notice also that since Λ is a diagonal matrix, it has at most N nonzero elements. Thus, we get lossless compression by at least a factor of N, and it will be greater than that if **A** has some zero (or negligible) singular values. Hence, the additional computation brings with it significant data compression.

Normally, several of the singular values are small enough to be ignored with little error. Thus "lossy" compression is achieved by ignoring the smaller $\Lambda_{i,i}$ values. The mean square error that results from this truncation is simply the sum of those singular values that are ignored.

The apparently miraculous compression power of the SVD transform is somewhat misleading. Although the entire image can be compressed into the diagonal elements of Λ, the kernel matrices **U** and **V** are unique for the image being compressed. These would have to be transmitted, along with the image, before reconstruction could occur at the receiving end. Possibly, however, one pair of kernal matrices could serve (approximately) for a group of similar images.

A Numerical Example. The SVD transform is illustrated in Figure 13–7, using a symmetric five-by-five pixel image.

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 2 & 1 & 0 \\ 1 & 3 & 4 & 3 & 1 \\ 2 & 4 & 5 & 4 & 2 \\ 1 & 3 & 4 & 3 & 1 \\ 0 & 1 & 2 & 1 & 0 \end{bmatrix} \quad \mathbf{A}\cdot\mathbf{A}^t = \begin{bmatrix} 6 & 14 & 18 & 14 & 6 \\ 14 & 36 & 48 & 36 & 14 \\ 18 & 48 & 65 & 48 & 18 \\ 14 & 36 & 48 & 36 & 14 \\ 6 & 14 & 18 & 14 & 6 \end{bmatrix} \quad \lambda = \begin{bmatrix} 147.07 \\ 1.872 \\ 0.058 \\ 0 \\ 0 \end{bmatrix} \quad \mathbf{U} = \begin{bmatrix} 0.186 & 0.638 & 0.241 & -0.695 & -0.695 \\ 0.476 & 0.058 & -0.52 & -0.133 & -0.128 \\ 0.691 & -0.422 & 0.587 & 0 & 0 \\ 0.476 & 0.058 & -0.52 & 0.133 & 0.128 \\ 0.186 & 0.638 & 0.241 & 0.695 & 0.695 \end{bmatrix}$$

$$\Lambda = \mathbf{U}^t\mathbf{A}\mathbf{U} = \begin{bmatrix} 12.585 & 0 & 0 & 0 & 0 \\ 0 & -1.142 & 0 & 0 & 0 \\ 0 & 0 & 0.557 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad \mathbf{A} = \mathbf{U}\Lambda\mathbf{U}^t = \begin{bmatrix} 0 & 1 & 2 & 1 & 0 \\ 1 & 3 & 4 & 3 & 1 \\ 2 & 4 & 6 & 4 & 2 \\ 1 & 3 & 4 & 3 & 1 \\ 0 & 1 & 2 & 1 & 0 \end{bmatrix}$$

Forward transform Inverse transform

Figure 13–7 The SVD transform of a symmetric five-by-five pixel image

13.7 TRANSFORM DOMAIN FILTERING

In Chapter 10, we saw that linear filtering—the action of a linear, shift-invariant system—can be modeled as a multiplication of the Fourier spectrum of an image by a transfer function defined in the frequency (i.e., transform) domain. While this important result is true only for the Fourier transform, analogous image-filtering operations can be defined for other transforms as well.

Like the Fourier transform, the general unitary transform expands an image as a weighted sum of basis images. The forward transformation process determines the weighting coefficients, while the inverse transformation reassembles the image from the expansion of the basis images.

Transform domain filtering involves modification of the weighting coefficients prior to reconstruction of the image via the inverse transform. With linear filtering, the transform is the Fourier transform, and the modification is effected by multiplying the spectrum by a transfer function. In the more general filtering case, the coefficient matrix is modified (by multiplication or other means) and the inverse transform produces the filtered image.

Clearly, it is the nature of the basis vectors (and of the resulting basis images) that establishes the different behavior of the various transforms. For example, sinusoidal noise contamination appears very compactly in the transform domain of a sinusoidal transform (recall Figure 10–8) and is thus easily removed by setting the corresponding coefficients to zero. The rectangular-wave transforms would be less well suited for this noise removal problem, since the contamination would not be as separable from the signal in their transform domains.

In general, if either the (desirable) signal components or the (undesirable) noise components of the image resemble one or a few of the basis images of a particular transform, then that transform will be useful in separating the two. This is because those components will be represented compactly in the transform domain. The general statement applies to problems of noise removal and signal detection as well.

The Haar transform, for example, is a good candidate for detecting vertical and horizontal lines and edges, since several of its basis images specifically match such features.

13.7.1 Edge, Line, and Spot Detection

Figure 13–8 illustrates the edge-detecting ability of the Haar transform on an eight-by-eight image. Since the transform is separable, an image feature that is a vertical or horizontal line

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \longrightarrow \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2.83 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Image Harr transform

Figure 13–8 Edge detection in an eight-by-eight image

or edge produces nonzero entries in only the first row and the first column of the transform image, respectively.

In the Haar transform, the feature produces at most $N/2$ non-zero entries. The position of the feature determines which (and how many) of the entries are nonzero. In the other transforms, all N entries of the first row or column are, in general, nonzero.

Figure 13–9 shows several transforms of an image containing a single-pixel spike (impulse). All N^2 elements of these transforms are nonzero except for those of the Haar transform, which has only $2N$ nonzero entries. Again, the location of the nonzero entries is determined by the position of the spike.

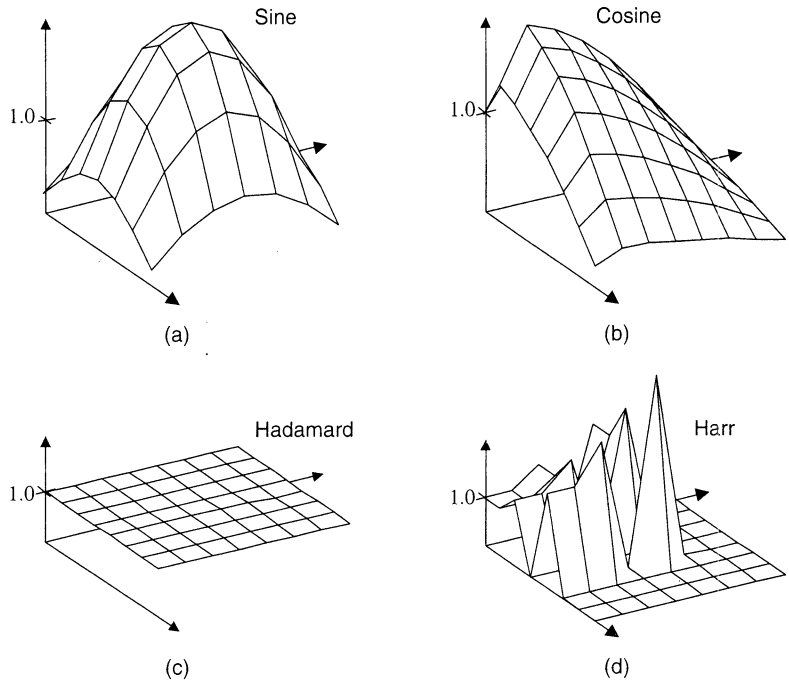

Figure 13–9 Transforms of an image containing an impulse: (a) DST; (b) DCT; (c) Hadamard: (d) Haar. The input is an eight-by-eight matrix, zero everywhere except the upper left element, which has value eight

13.7.2 Filter Design

Because of its close association with shift-invariant linear systems, the Fourier transform has a well-developed theoretical background to guide its use in image-filtering applications. The other transforms are less well supported in theory, and their use is often more experimental. An understanding of the similarities and differences among these transforms helps guide the search for workable solutions.

13.8 SUMMARY OF IMPORTANT POINTS

1. The rows of an N-by-N transformation matrix are a set or orthonormal basis vectors for an N-dimensional vector space.

2. A unitary linear transformation generates a vector of *N transform coefficients,* each of which is the inner product of the input vector with one of the rows of the transform matrix.
3. The inverse transform is formed similarly, by inner products of the transform coefficient vector with the rows of the inverse transform matrix.
4. The inverse transform can also be viewed as forming a weighted summation of the basis vectors, where the transform coefficients are the weights.
5. For a two-dimensional symmetric, separable unitary transformation, the basis images are the outer products of the rows of the transform matrix.

PROBLEMS

1. The eigenvalues of an eight-channel multispectral image are [6.1 168 0.08 13 64 214 1.2 0.2]. What will be the RMS error if you use principal component analysis for 2:1 data compression?
2. Design an 8 by 8 Haar transform filter mask that will remove small horizontal edges from an image.

PROJECTS

1. Develop a program that implements the discrete cosine transform, and use the program to demonstrate highpass filtering for image enhancement.
2. Develop a program that implements the discrete Hartley transform, and use the program to demonstrate lowpass filtering for noise reduction.
3. Develop a program that uses principal-component analysis to reduce a 24-bit-per-pixel color image to a 16-bit-per-pixel representation and back. Produce demonstration images, and comment upon the resulting degradation.
4. Develop a program that implements the slant transform, and use the program to demonstrate the removal of linear shading.
5. Develop a program that implements the Haar transform, and use the program to show the edges in an image.

REFERENCES

For further reading on topics relating to image transforms, see Appendix 2.

1. N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete Cosine Transforms," *IEEE Trans. Comp.,* **C-23**:90–93, 1974.
2. M. J. Narasimha and A. M. Peterson, "On the Computation of the Discrete Cosine Transform," *IEEE Trans. Commun.,* **COM-26,** 6:934–936, 1978.
3. W. H. Chen, C. H. Smith, and S. C. Fralick, "A Fast Computational Algorithm for the Discrete Cosine Transform," *IEEE Trans. Commun.,* **COM-25,** 1004–1009, 1977.
4. A. K. Jain, "A Fast Karhunen-Loéve Transforms for a Class of Random Processes," *IEEE Trans. Commun.,* **COM-24,** 1023–1029, 1975.
5. A. K. Jain, *Fundamentals of Digital Image Processing,* Prentice-Hall, Englewood Cliffs, NJ, 1989.
6. P. Yip and K. R. Rao, "A Fast Computational Algorithm for the Discrete Sine Transform," *IEEE Trans. Commun.,* **COM-28**(2), 304–307, 1980.

7. R. V. L. Hartley, "A More Symmetrical Fourier Analysis Applied to Transmission Problems," *Proc. IRE,* **30:**144–150, 1942.
8. R. M. Bracewell, "The Discrete Hartley Transform," *J. Opt. Soc. Am.,* **73**(12):1832–1835, 1983.
9. R. M. Bracewell, *The Fourier Transform and Its Applications* (2d ed.), McGraw-Hill, New York, 1986.
10. A. K. Jain, "A Sinusoidal Family of Unitary Transforms," *IEEE Trans. Pattern Anal. Machine Intell.,* **PAMI-1**(4):356–365, 1979.
11. J. Hadamard, "Resolution d'une Question Relative aux Determinants," *Bull. Sci. Math.,* Ser. 2, **17,** Part I, 240–246, 1893.
12. J. E. Whelchel, Jr., and D. F. Guinn, "The Fast Fourier-Hadamard Transform and its Use in Signal Representation and Classification," *EASCON 1968 Convention Record,* 561–573, 1968.
13. W. K. Pratt, H. C. Andrews, and J. Kane, "Hadamard Transform Image Coding," *Proc. IEEE,* **57**(1):58–68, 1969.
14. H. Kitajima, "Energy Packing Efficiency of the Hadamard Transform," *IEEE Trans. Comm.* (correspondence), **COM-24,** 1256–1258.
15. J. Williamson, "Hadamard's Determinant Theorem and the Sum of Four Squares," *Duke Math. J.,* **11:**65–81, 1944.
16. H. F. Harmuth, "A Generalization Concept of Frequency and Some Applications," *IEEE Trans. Info. Theory,* **IT-14**(3):375–382, 1968.
17. J. L. Walsh, "A Closed Set of Normal Orthogonal Functions," *Am. J. Math.,* **45**(1):5–24, 1923.
18. W. K. Pratt, W. H. Chen, and L. R. Welch, "Slant Transform Image Coding," *IEEE Trans. Comm.,* **COM-22**(8):1075–1093, 1974.
19. A. Haar, "Zur Theorie der Orthogonalen Funktionen-System," Inaugural Dissertation, *Math. Annalen,* **5**:17–31, 1955.
20. H. C. Andrews, *Computer Techniques in Image Processing,* Academic Press, New York, 1970.
21. J. E. Shore, "On the Application of Haar Functions," *IEEE Trans. Comm.,* **COM-21,** 209–216, 1973.
22. H. Hotelling, "Analysis of a Complex of Statistical Variables into Principal Components," *J. Educ. Psychol.,* 24, 417–441, 498–520, 1933.
23. K. Karhunen, "Über Lineare Methoden in der Wahrscheinlich-Keitsrechnung," *Ann. Acad. Sci. Fennicae,* Ser. A.I.37 (English translation by I. Selin, "On Linear Methods in Probability Theory," Doc. T-131, The RAND Corp., Santa Monica, CA, 1960), 1947.
24. M. Loéve, "Fonctions Aléatoires de Second Ordre," in P. Lévy, *Processus Stochastiques et Mouvement Brownien,* Hermann, Paris, 1948.
25. W. K. Pratt, *Digital Image Processing* (2d ed.), John Wiley & Sons, New York, 1991.
26. R. J. Clark, *Transform Coding of Images,* Academic Press, New York, 1985.
27. G. H. Golub and C. Reinsch, "Singular Value Decomposition and Least Squares Solutions," *Numer. Math.,* **14,** 403–420, 1970.