FFT — REAL DATA

① set $z = x + i.0$
do FFT

wasteful

② easy way
If one needs to FFTs x, y

Let

$$z_j = x_j + i y_j$$

Then

$$\hat{x}_k = \frac{\hat{z}_k + \hat{z}_{N-k}}{2} \qquad \hat{y}_k = \frac{\hat{z}_k - \hat{z}_{N-k}}{2}$$

9 Inversion of the FFT of a real sequence

In this section we show how the symmetry property (5.6) of the FFT of a real sequence can be used to reduce by about one-half the computations involved in inverting the FFT. Although there are other methods of doing this than the method described below, this method is of interest because it uses no extra sines other than the set $\{S(m)\}_{m=0}^{(1/4)N}$ that was used for performing the FFT itself.

We will assume in this section that $W = e^{i2\pi/N}$. Let R_k and I_k stand for the real and imaginary parts of the FFT $\{F_k\}$. That is,

$$F_k = R_k + iI_k, \qquad (k = 0, 1, \ldots, N-1) \tag{9.1}$$

where F_k is defined by

$$F_k = \sum_{j=0}^{N-1} f_j W^{jk}.$$

Then, by the formula for DFT inversion,

$$f_j = \frac{1}{N} g_j, \qquad (j = 0, 1, \ldots, N-1) \tag{9.2}$$

where

$$g_j = \sum_{k=0}^{N-1} F_k (W^{jk})^*. \tag{9.3}$$

Using $W = e^{i2\pi/N}$ and (9.1) we have

$$\begin{aligned} g_j &= \sum_{j=0}^{N-1} (R_k + iI_k) e^{-i2\pi jk/N} \\ &= \sum_{k=0}^{N-1} (R_k + iI_k) \left(\cos \frac{2\pi jk}{N} - i \sin \frac{2\pi jk}{N} \right). \end{aligned} \tag{9.4}$$

Since $\{f_j\}$ consists of real numbers, so does $\{g_j\}$ because of (9.2). Consequently, only the real part of the sum in (9.4) is nonzero. Therefore, we must have

$$g_j = \sum_{k=0}^{N-1} R_k \cos \frac{2\pi jk}{N} + \sum_{k=0}^{N-1} I_k \sin \frac{2\pi jk}{N}. \tag{9.5}$$

Because of (5.6) we have

$$R_{N-k} = R_k, \qquad I_{N-k} = -I_k, \qquad (k = 1, \ldots, N-1). \tag{9.6}$$

It follows from (9.6) that, for each fixed j, the sequences

$$\left\{ R_k \cos \frac{2\pi jk}{N} \right\}_{k=1}^{N-1} \qquad \text{and} \qquad \left\{ I_k \sin \frac{2\pi jk}{N} \right\}_{k=1}^{N-1}$$

are even about $(1/2)N$. From formula (7.3) we obtain

$$\sum_{k=1}^{N-1} R_k \cos \frac{2\pi jk}{N} = R_{\frac{1}{2}N}(-1)^j + 2 \sum_{k=1}^{\frac{1}{2}N-1} R_k \cos \frac{2\pi jk}{N} \tag{9.7}$$

and

$$\sum_{k=1}^{N-1} I_k \sin \frac{2\pi jk}{N} = 2 \sum_{k=1}^{\frac{1}{2}N-1} I_k \sin \frac{2\pi jk}{N}. \tag{9.8}$$

Using (9.7) and (9.8) in (9.5) yields

$$g_j = R_0 + R_{\frac{1}{2}N}(-1)^j + 2 \sum_{k=1}^{\frac{1}{2}N-1} R_k \cos \frac{\pi jk}{\frac{1}{2}N} + 2 \sum_{k=1}^{\frac{1}{2}N-1} I_k \sin \frac{\pi jk}{\frac{1}{2}N}. \tag{9.9}$$

Formula (9.9) shows that $\{g_j\}_{j=0}^{N-1}$, and consequently $\{f_j\}_{j=0}^{N-1}$, can be generated from a $(1/2)N$-point fast cosine transform of

$$\{0,\ 2R_1,\ 2R_2, \ldots,\ 2R_{\frac{1}{2}N-1}\}$$

and a fast sine transform of

$$\{2I_1,\ 2I_2, \ldots,\ 2I_{\frac{1}{2}N-1}\}.$$

These fast cosine and fast sine transforms are even and odd about $(1/2)N$, respectively. Taking this symmetry into account, formula (9.9) applies to $j = 0$, $1, \ldots, N-1$.

As we noted above, an interesting feature of (9.9) is that to compute the fast cosine and fast sine transforms, *of order* $(1/2)N$, by the methods of Section 7, one needs only the *same* sines $\{S(m)\}_{m=0}^{(1/4)N}$ generated to calculate the FFT $\{F_k\}_{k=0}^{N-1}$. Therefore, inverting the FFT using (9.9) requires only these same sines. This is a useful memory savings.

Cosine Transform

Define

$$\alpha_0 = \sqrt{\frac{1}{N}}$$

$$\alpha_k = \sqrt{\frac{2}{N}}\; k = 1, 2, ..., N-1$$

Then

$$v_k = \alpha_k \sum_{n=0}^{N-1} u_n \cos\left[\frac{(2n+1)k\pi}{2N}\right] k = 0, 1, ...N-1$$

Note this uses **half** points in the interval.
Then the inverse is given by

$$u_n = \sum_{k=0}^{N-1} \alpha_k v_k \cos\left[\frac{(2n+1)k\pi}{2N}\right]\; n = 0, 1, ...N-1$$

To evaluate this fast we relate it to a FFT

$$v_k = \mathrm{Re}\left\{\alpha_k e^{-\frac{\pi i k}{2N}} \sum_{n=0}^{N-1} \tilde{u}_n e^{-\frac{2\pi i k n}{N}}\right\} = \mathrm{Re}\left\{\alpha_k w_{2N}^k DFT(u)\right\}$$

where

$$\tilde{u}_n = u_{2n}$$

$$\tilde{u}_{N-n-1} = u_{2n} + 1 \quad 0 \le n \le \frac{N}{2} - 1$$

Inverse

$$u_n = \sum_{k=0}^{N-1} \alpha_k v_k \cos\left[\frac{(2n+1)k\pi}{2N}\right] \; n = 0,1,\ldots N-1$$

Define

$$\tilde{u}_{2n} = \mathrm{Re}\left\{\left[\sum_{k=0}^{N-1} \alpha_k v_k e^{\frac{k\pi i}{2N}}\right] e^{\frac{2\pi i k n}{N}}\right\} \quad n = 0,1,\ldots \frac{N}{2} - 1$$

Then

$$u_{2n} = \tilde{u}_{2n}$$

$$u_{2n+1} = \tilde{u}_{2(N-1-n)}$$

then

$$\hat{u}_k = \alpha_k \left\{ \sum_{n=0}^{\frac{N}{2}-1} u_{2n} \cos\left[\frac{(4n+1)k\pi}{2N}\right] + u_{2n+1} \cos\left[\frac{(4n+3)k\pi}{2N}\right] \right\}$$

$$= \alpha_k \left\{ \sum_{n=0}^{\frac{N}{2}-1} \bar{u}_n \cos\left[\frac{(4n+1)k\pi}{2N}\right] + \bar{u}_{N-n-1} \cos\left[\frac{(4n+3)k\pi}{2N}\right] \right\}$$

let $n' = N-n-1$

$\Longrightarrow \quad n = N-n'-1$

$$= \alpha_k \left\{ \sum_{n=0}^{\frac{N}{2}-1} \bar{u}_n \cos\left[\frac{(4n+1)k\pi}{2N}\right] + \sum_{n'=\frac{N}{2}}^{N-1} \bar{u}_{n'} \cos\left[\frac{(4(N-n')-1)k\pi}{2N}\right]\right.$$

$$= \alpha_k \sum_{n=0}^{N-1} \bar{u}_n \cos\left[\frac{(4n+1)k\pi}{2N}\right]$$

$$= \mathrm{Re}\left\{ \alpha_k e^{-\frac{\pi k i}{2N}} \sum_{n=0}^{N-1} \bar{u}_n e^{-\frac{2\pi i k n}{N}} \right\}$$

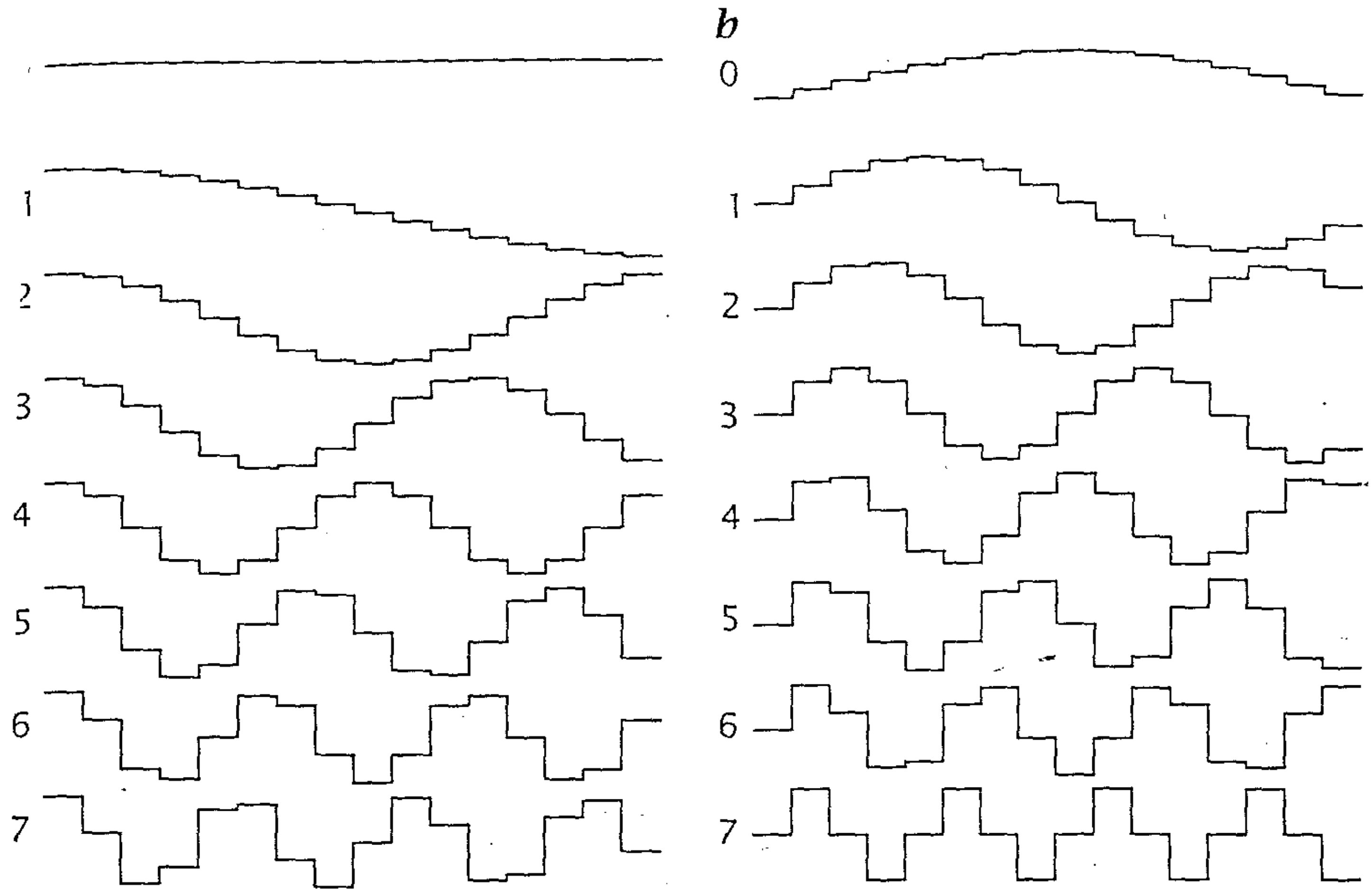
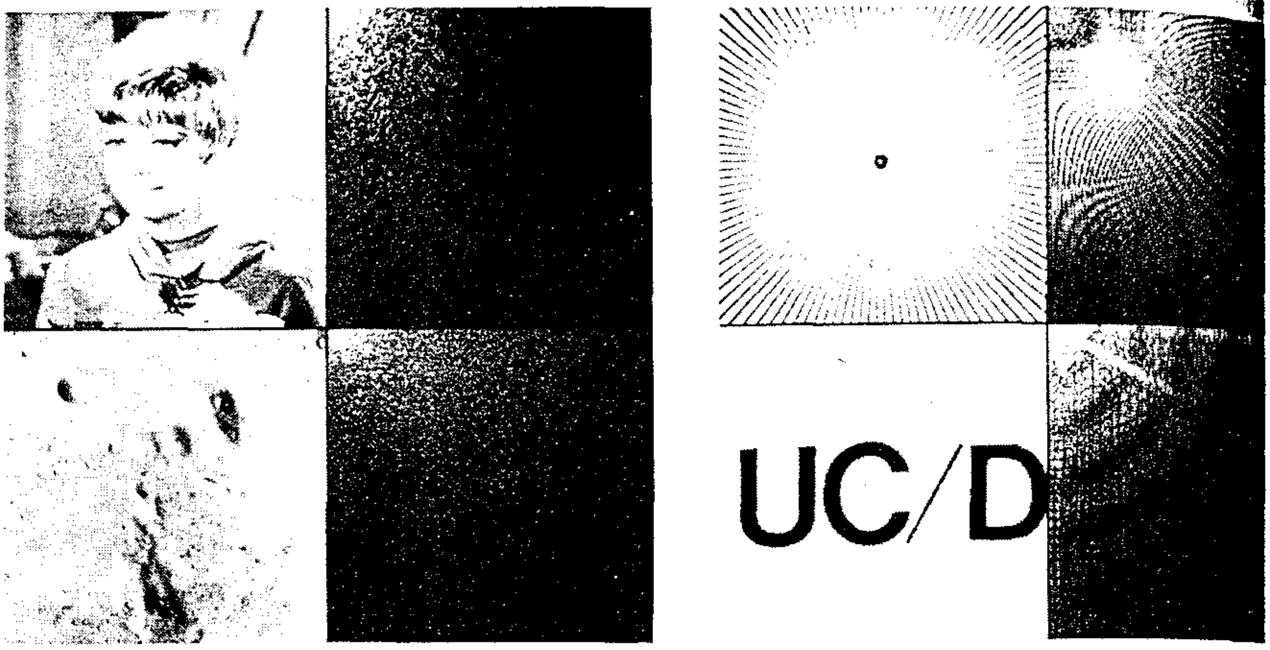


*Figure 2.18: First 8 base functions of one-dimensional unitary transforms for $N = 16$: **a** cosine transform and **b** sine transform.*

The cosine and sine functions only span the subspaces of the even and odd functions, respectively. Basis vectors with the missing symmetry can be generated, however, if trigonometric functions with half-integer wavelengths are added. This is equivalent to doubling the base wavelength. Consequently, the kernels for the *cosine* and *sine transforms* in an M-dimensional vector space are

$$
\begin{aligned}
C_{nv} &= \cos\left(\frac{\pi n v}{N}\right), \\
S_{nv} &= \sin\left(\frac{\pi n (v+1)}{N}\right).
\end{aligned}
\qquad (2.44)
$$

... of the 1-D cosine and sine func-



(a) Cosine transform examples of monochrome images;

(b) Cosine transform examples of binary im

Figure 5.11

3. The cosine transform is a fast transform. The cosine transform of a vecto elements can be calculated in $O(N \log_2 N)$ operations via an N-poin [19]. To show this we define a new sequence $\tilde{u}(n)$ by reordering the eve odd elements of $u(n)$ as

$$\left.\begin{aligned}\tilde{u}(n) &= u(2n)\\ \tilde{u}(N-n-1) &= u(2n+1)\end{aligned}\right\}, \qquad 0 \le n \le \left(\frac{N}{2}\right) - 1$$

Now, we split the summation term in (5.87) into even and odd terms a (5.91) to obtain

$$\begin{aligned} v(k) &= \alpha(k)\left\{\sum_{n=0}^{(N/2)-1} u(2n) \cos\left[\frac{\pi(4n+1)k}{2N}\right]\right. \\ &\quad \left. + \sum_{n=0}^{(N/2)-1} u(2n+1) \cos\left[\frac{\pi(4n+3)k}{2N}\right]\right\} \\ &= \alpha(k)\left\{\left[\sum_{n=0}^{(N/2)-1} \tilde{u}(n) \cos\left[\frac{\pi(4n+1)k}{2N}\right]\right.\right. \\ &\quad \left. + \sum_{n=0}^{(N/2)-1} \tilde{u}(N-n-1) \cos\left[\frac{\pi(4n+3)k}{2N}\right]\right\} \end{aligned}$$

Changing the index of summation in the second term to $n' = N - n -$ combining terms, we obtain

e Cosine Transform

The dct2 function in the Image Processing Toolbox computes the two-dimensional discrete cosine transform (DCT) of an image. The DCT has the property that, for a typical image, most of the visually significant information about the image is concentrated in just a few coefficients of the DCT. For this reason, the DCT is often used in image compression applications. For example, the DCT is at the heart of the the international standard lossy image compression algorithm known as JPEG. (The name comes from the working group that developed the standard: the Joint Photographic Experts Group.)

The two-dimensional DCT of an M-by-N matrix A is defined as follows:

$$B_{pq} = \alpha_p \alpha_q \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} A_{mn} \cos\frac{\pi(2m+1)p}{2M} \cos\frac{\pi(2n+1)q}{2N}, \quad \begin{matrix} 0 \le p \le M-1 \\ 0 \le q \le N-1 \end{matrix}$$

$$\alpha_p = \begin{cases} 1/\sqrt{M}, & p = 0 \\ \sqrt{2/M}, & 1 \le p \le M-1 \end{cases} \qquad \alpha_q = \begin{cases} 1/\sqrt{N}, & q = 0 \\ \sqrt{2/N}, & 1 \le q \le N-1 \end{cases}$$

The values B_{pq} are called the *DCT coefficients* of A. (Note that matrix indices in MATLAB always start at 1 rather than 0; therefore, the MATLAB matrix elements A(1,1) and B(1,1) correspond to the mathematical quantities A_{00} and B_{00}, respectively.)

The DCT is an invertible transform, and its inverse is given by:

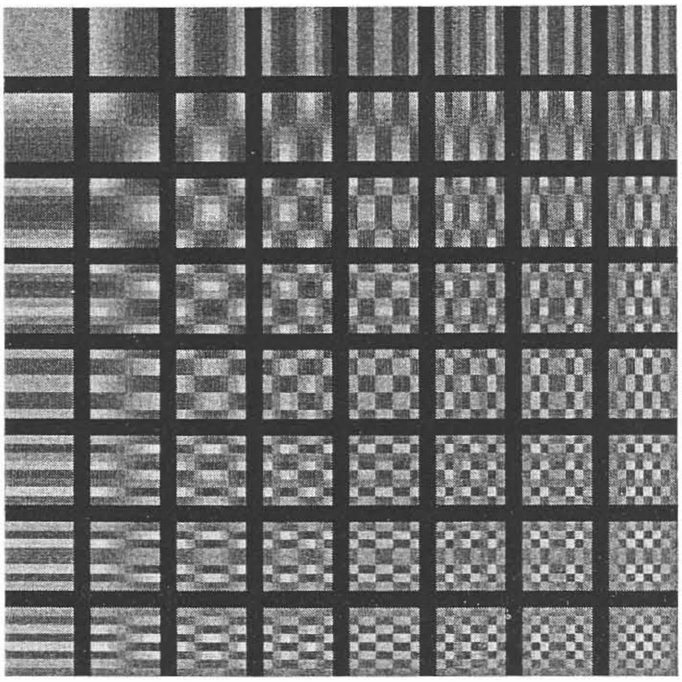
$$A_{mn} = \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} \alpha_p \alpha_q B_{pq} \cos\frac{\pi(2m+1)p}{2M} \cos\frac{\pi(2n+1)q}{2N}, \quad \begin{matrix} 0 \le m \le M-1 \\ 0 \le n \le N-1 \end{matrix}$$

$$\alpha_p = \begin{cases} 1/\sqrt{M}, & p = 0 \\ \sqrt{2/M}, & 1 \le p \le M-1 \end{cases} \qquad \alpha_q = \begin{cases} 1/\sqrt{N}, & q = 0 \\ \sqrt{2/N}, & 1 \le q \le N-1 \end{cases}$$

The inverse DCT equation can be interpreted as meaning that any M-by-N matrix A can be written as a sum of MN functions of the form:

$$\alpha_p \alpha_q \cos\frac{\pi(2m+1)p}{2M} \cos\frac{\pi(2n+1)q}{2N}, \quad \begin{matrix} 0 \le p \le M-1 \\ 0 \le q \le N-1 \end{matrix}$$

These functions are called the *basis functions* of the DCT. The DCT coefficients B_{pq}, then, can be regarded as the *weights* applied to each basis function. For 8-by-8 matrices, the 64 basis functions are illustrated by this image:



Horizontal frequencies increase from left to right, and vertical frequencies increase from top to bottom. The constant-valued basis function at the upper left is often called the *DC basis function*, and the corresponding DCT coefficient B_{00} is often called the *DC coefficient*.

The DCT Transform Matrix

The Image Processing Toolbox offers two different ways to compute the DCT. The first method is to use the function `dct2`. `dct2` uses an FFT-based algorithm for speedy computation with large inputs.

For small square inputs, such as 8-by-8 or 16-by-16, it may be more efficient to use the DCT *transform matrix*, which is returned by the function `dctmtx`. The M-by-M transform matrix `T` is given by:

$$T_{pq} = \begin{cases} \dfrac{1}{\sqrt{M}} & p = 0, \quad 0 \le q \le M-1 \\ \sqrt{\dfrac{2}{M}} \cos \dfrac{\pi(2q+1)p}{2M} & 1 \le p \le M-1, \quad 0 \le q \le M-1 \end{cases}$$

For an M-by-M matrix `A`, `T*A` is an M-by-M matrix whose columns contain the one-dimensional DCT of the columns of `A`. The two-dimensional DCT of `A` can be computed as `B=T*A*T'`. Since `T` is a real orthonormal matrix, its inverse is the same as its transpose. Therefore, the inverse two-dimensional DCT of `B` is given by `T'*B*T`.

The DCT and Image Compression

In the JPEG image compression algorithm, the input image is divided into 8-by-8 or 16-by-16 blocks, and the two-dimensional DCT is computed for each block. The DCT coefficients are then quantized, coded, and transmitted. The JPEG receiver (or JPEG file reader) decodes the quantized DCT coefficients, computes the inverse two-dimensional DCT of each block, and then puts the blocks back together into a single image. For typical images, many of the DCT coefficients have values close to zero; these coefficients can be discarded without seriously affecting the quality of the reconstructed image.

The example code below computes the two-dimensional DCT of 8-by-8 blocks in the input image; discards (sets to zero) all but 10 of the 64 DCT coefficients in each block; and then reconstructs the image using the two-dimensional inverse DCT of each block. The transform matrix computation method is used.

```
I = imread('cameraman.tif');
I = double(I)/255;
T = dctmtx(8);
B = blkproc(I,[8 8],'P1*x*P2',T,T');
mask = [1   1   1   1   0   0   0   0
        1   1   1   0   0   0   0   0
        1   1   0   0   0   0   0   0
        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];
B2 = blkproc(B,[8 8],'P1.*x',mask);
I2 = blkproc(B2,[8 8],'P1*x*P2',T',T);
imshow(I), figure, imshow(I2)
```

Although there is some loss of quality in the reconstructed image, it is clearly recognizable, even though almost 85% of the DCT coefficients were discarded. To experiment with discarding more or fewer coefficients, and to apply this technique to other images, try running the demo function `dctdemo`.

To derive the inverse cosine transform relation, we relate $C_x(\omega_1, \omega_2)$ to $R(\omega_1, \omega_2)$ using (3.57), relate $R(\omega_1, \omega_2)$ to $r(n_1, n_2)$ using the inverse Fourier transform relation, and then relate $r(n_1, n_2)$ to $x(n_1, n_2)$ using (3.56). The result is

$$x(n_1, n_2) = \frac{1}{(2\pi)^2}\int_{\omega_1=-\pi}^{\pi}\int_{\omega_2=-\pi}^{\pi} C_x(\omega_1, \omega_2)\cos\omega_1(n_1+\tfrac{1}{2})\cos\omega_2(n_2+\tfrac{1}{2})\,d\omega_1\,d\omega_2,$$

$$n_1 \geq 0, n_2 \geq 0. \qquad (3.60)$$

From (3.58) and (3.60)

Cosine Transform Pair

$$C_x(\omega_1, \omega_2) = \sum_{n_1=0}^{\infty}\sum_{n_2=0}^{\infty} 4x(n_1, n_2)\cos\omega_1(n_1+\tfrac{1}{2})\cos\omega_2(n_2+\tfrac{1}{2}). \qquad (3.61a)$$

$$x(n_1, n_2) = \begin{cases} \dfrac{1}{(2\pi)^2}\displaystyle\int_{\omega_1=-\pi}^{\pi}\int_{\omega_2=-\pi}^{\pi} C_x(\omega_1, \omega_2)\cos\omega_1(n_1+\tfrac{1}{2})\cos\omega_2(n_2+\tfrac{1}{2})\,d\omega_1\,d\omega_2 & \\ \qquad n_1 \geq 0,\ n_2 \geq 0 & (3.61b) \\ 0, \qquad \text{otherwise.} & \end{cases}$$

Many properties of the cosine transform can be derived from (3.61), or (3.55) and the Fourier transform properties. Some of the more important properties are listed in Table 3.4. From the symmetry properties, $C_x(\omega_1, \omega_2)$ is an even function and in addition is symmetric with respect to the ω_1 and ω_2 axes. When $x(n_1, n_2)$ is real, its cosine transform $C_x(\omega_1, \omega_2)$ is also real.

TABLE 3.4 PROPERTIES OF THE COSINE TRANSFORM

$x(n_1, n_2), y(n_1, n_2) = 0$ outside $n_1 \geq 0, n_2 \geq 0$

$x(n_1, n_2) \longleftrightarrow C_x(\omega_1, \omega_2)$

$y(n_1, n_2) \longleftrightarrow C_y(\omega_1, \omega_2)$

Property 1. *Linearity*

$$ax(n_1, n_2) + by(n_1, n_2) \longleftrightarrow aC_x(\omega_1, \omega_2) + bC_y(\omega_1, \omega_2)$$

Property 2. *Separable Sequence*

$$x(n_1, n_2) = x_1(n_1)x_2(n_2) \longleftrightarrow C_x(\omega_1, \omega_2) = C_{x1}(\omega_1)C_{x2}(\omega_2)$$

Property 3. *Energy Relationship*

$$\sum_{n_1=0}^{\infty}\sum_{n_2=0}^{\infty}|x(n_1, n_2)|^2 = \frac{1}{4(2\pi)^2}\int_{\omega_1=-\pi}^{\pi}\int_{\omega_2=-\pi}^{\pi}|C_x(\omega_1, \omega_2)|^2\,d\omega_1\,d\omega_2$$

Property 4. *Symmetry Properties*

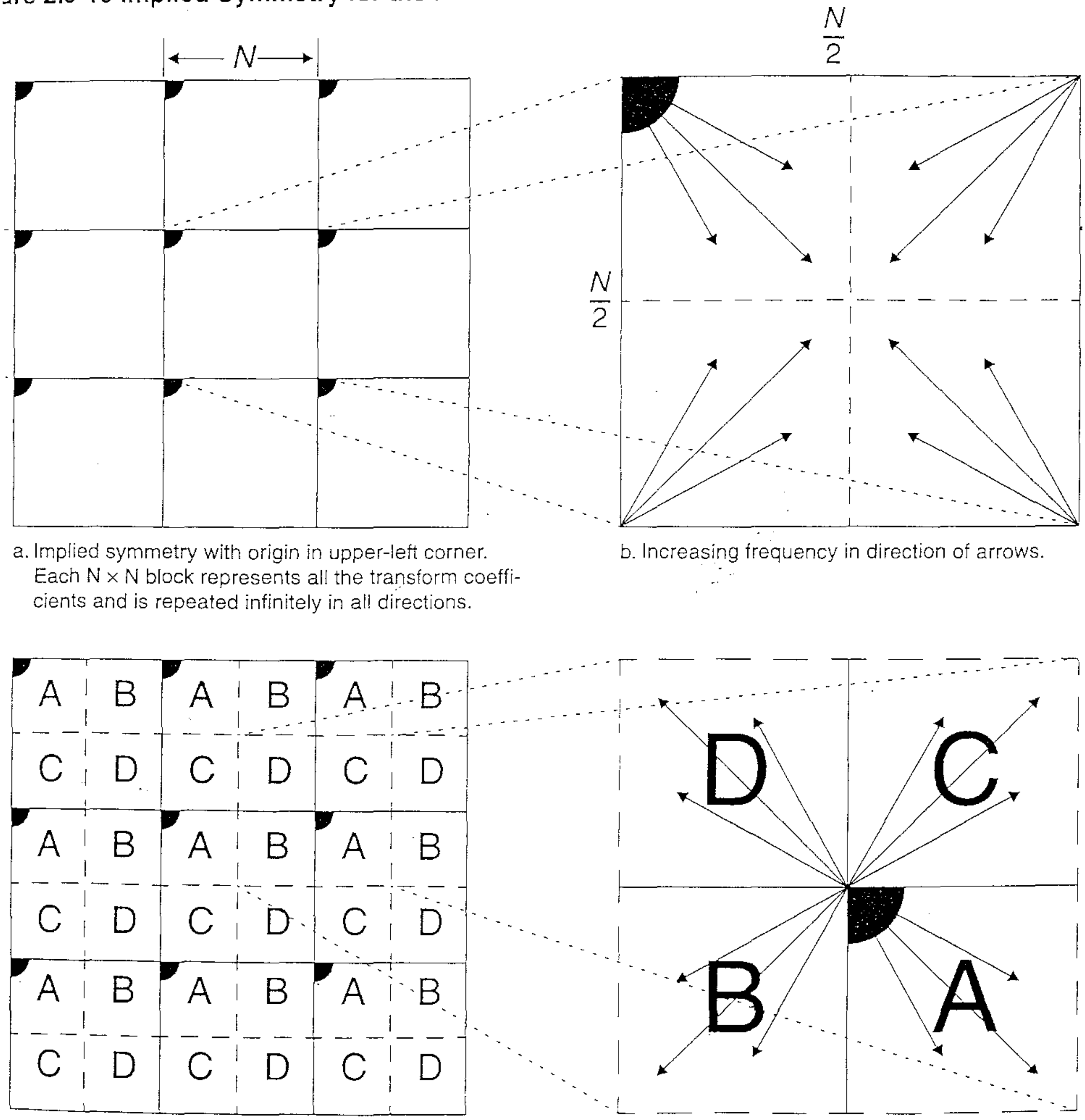
(a) $C_x(\omega_1, \omega_2) = C_x(-\omega_1, \omega_2) = C_x(\omega_1, -\omega_2) = C_x(-\omega_1, -\omega_2)$

(b) $x^*(n_1, n_2) \longleftrightarrow C_x^*(\omega_1, \omega_2)$

(c) real $x(n_1, n_2) \longleftrightarrow$ real $C_x(\omega_1, \omega_2)$

quency information. They are used for image compression or for hiding effects caused by noise. Visually they blur the image, although this blur is sometimes considered an enhancement because it imparts a softer effect to the image (see Figure 2.5-17). Lowpass filtering is performed by multiplying the spectrum by a filter and then applying the inverse transform to obtain the filtered image. The ideal filter function is shown in

ure 2.5-13 Implied Symmetry for the Fourier Transform



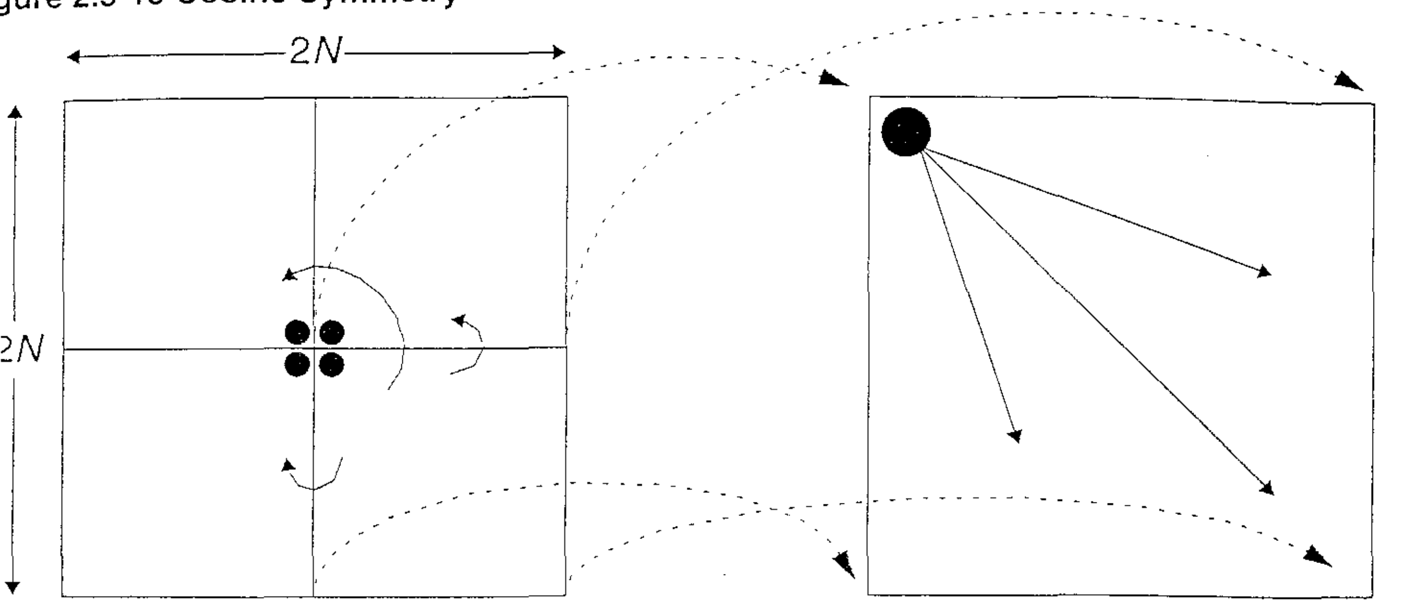
a. Implied symmetry with origin in upper-left corner. Each N × N block represents all the transform coefficients and is repeated infinitely in all directions.

b. Increasing frequency in direction of arrows.

c. Periodic spectrum, with quadrants labeled A,B,C,D.

d. Spectrum shifted to center. Frequency increases in all directions as we move away from the origin.

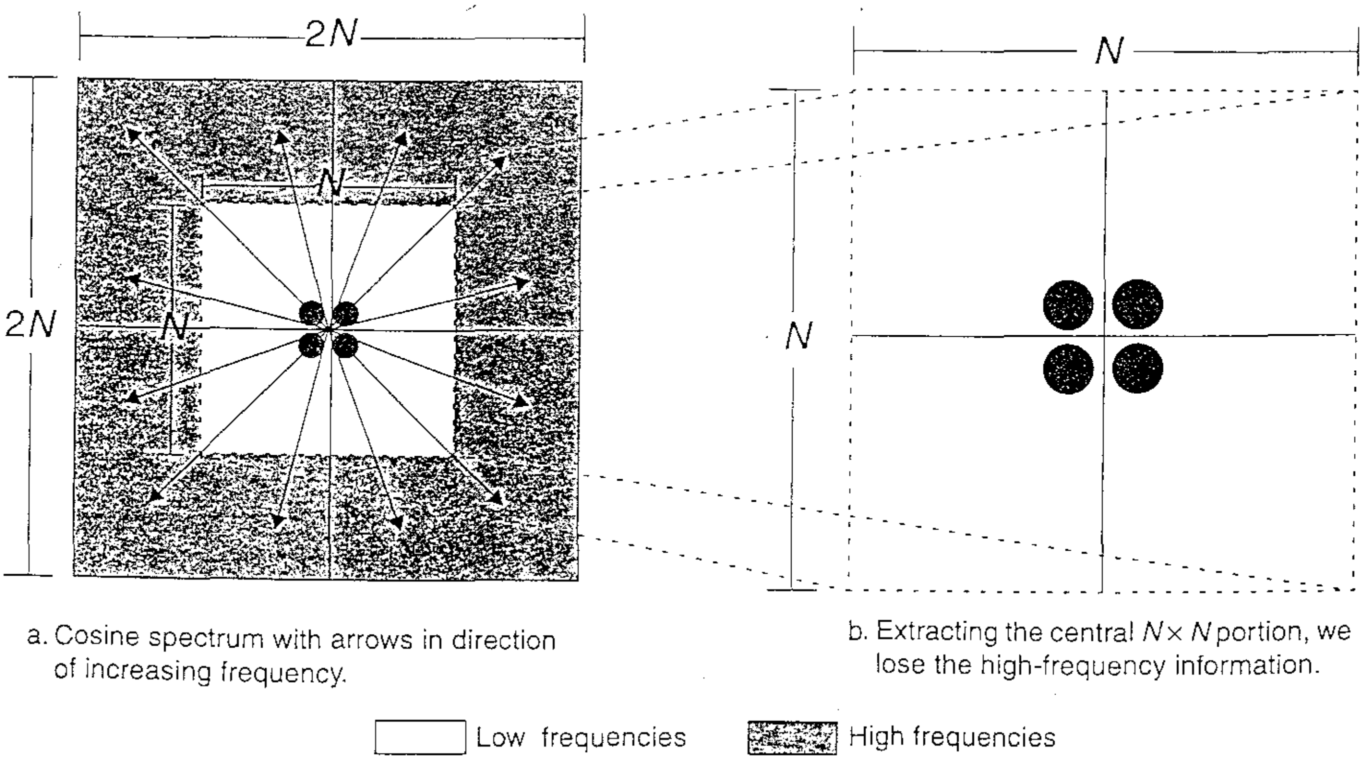
gure 2.5-15 Cosine Symmetry



a. Spectrum folded about origin, represented by the ●. The $2N \times 2N$ block is repeated infinitely in all directions.

b. Arrows indicate direction of increasing frequency for cosine spectrum.

jure 2.5-16 Cosine Spectrum Should Not Be Shifted to Center



a. Cosine spectrum with arrows in direction of increasing frequency.

b. Extracting the central $N \times N$ portion, we lose the high-frequency information.

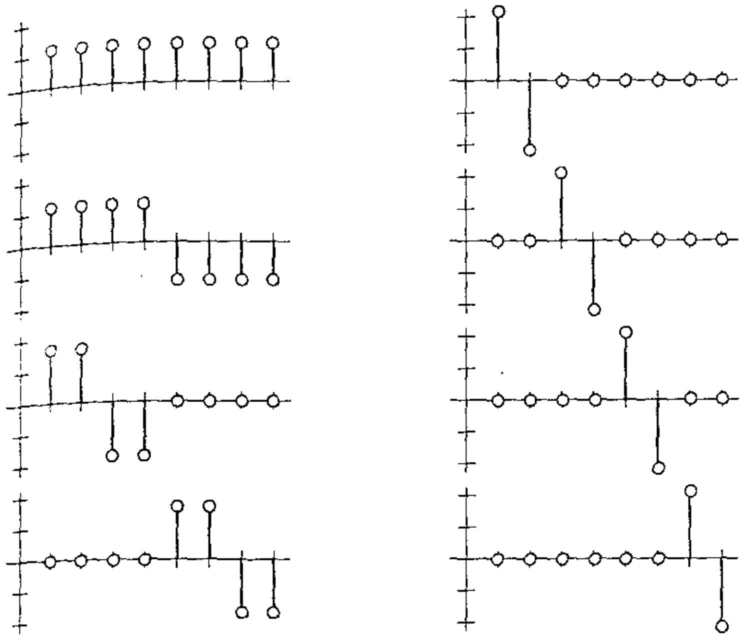


Figure 13–5 The Haar transform basis functions for $N = 8$

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

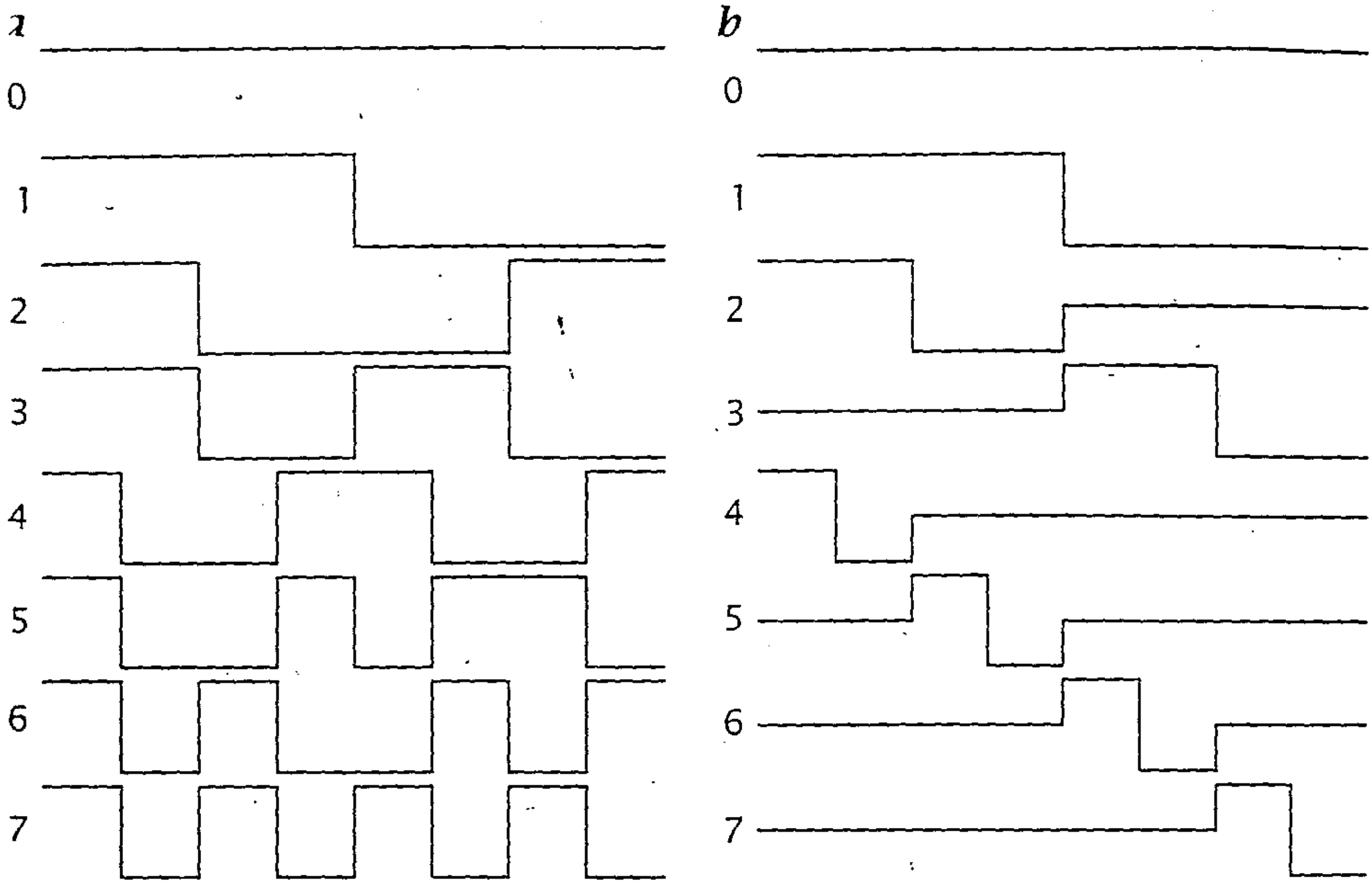


Figure 2.19: *First 8 base functions of one-dimensional unitary transforms for $N = 16$:* ***a*** *Hadamard transform and* ***b*** *Haar transform;*

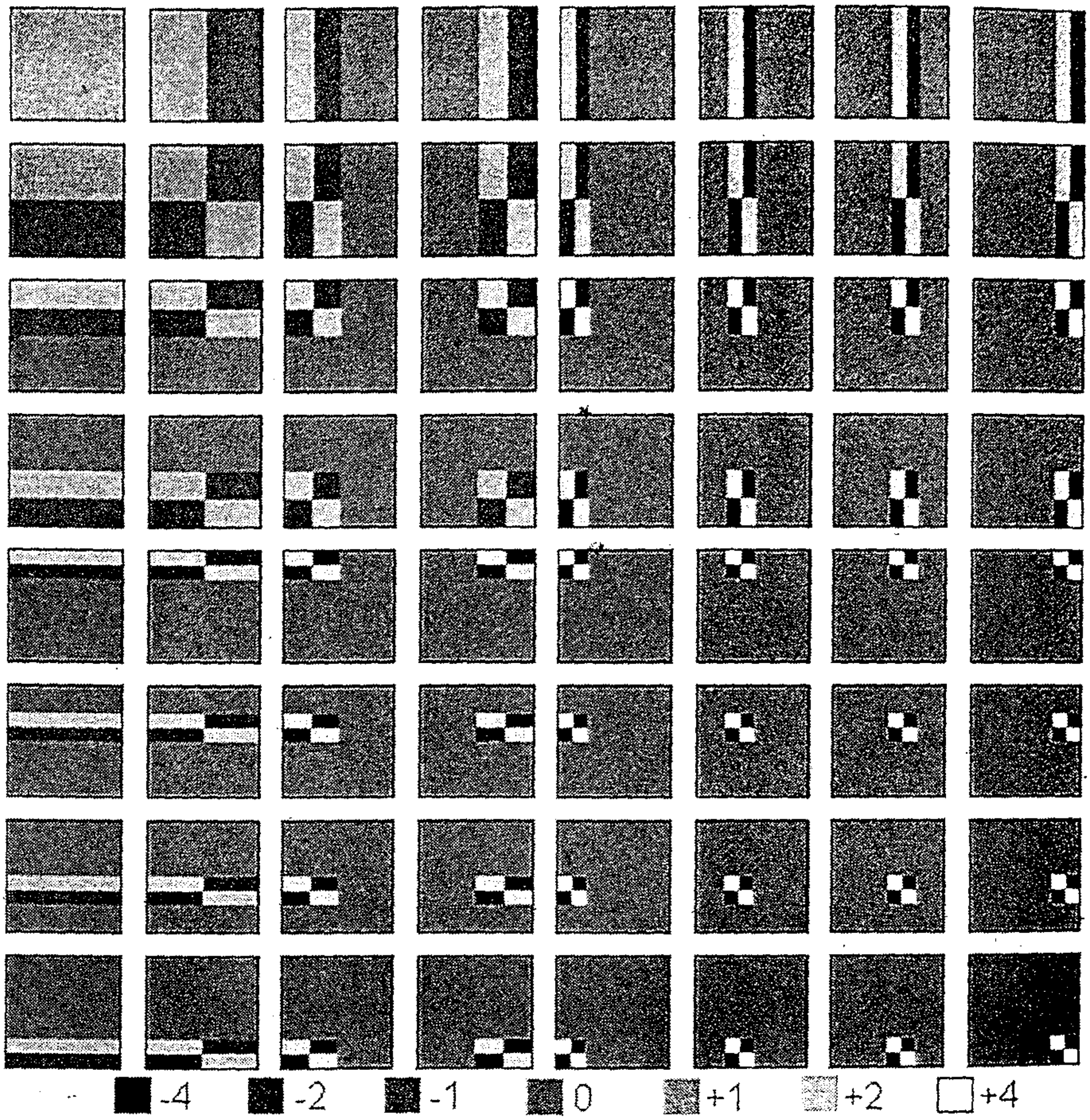


Figure 13–6 The Haar transform basis images for $N = 8$

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

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

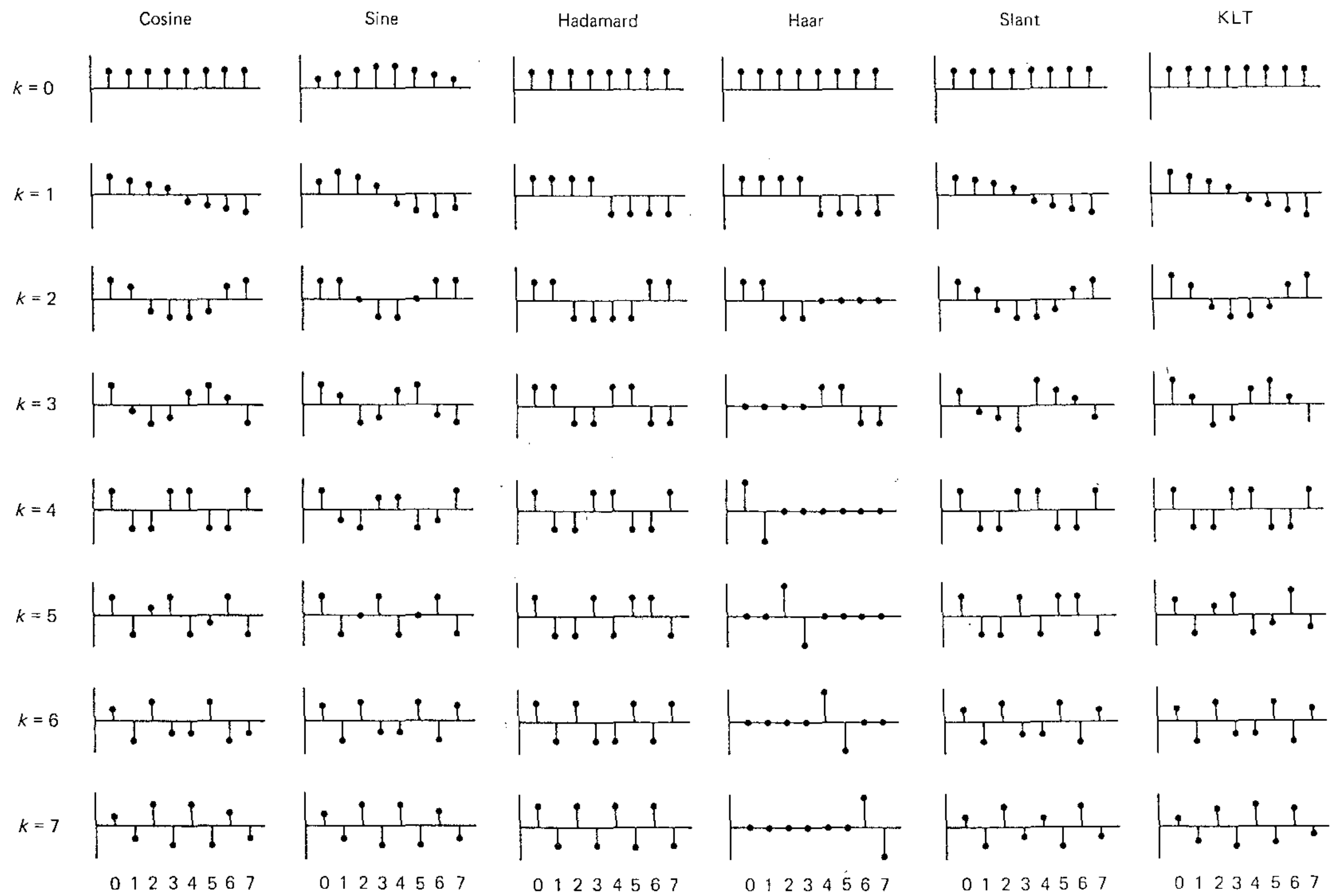


Figure 5.1 Basic vectors of the 8 × 8 transforms.

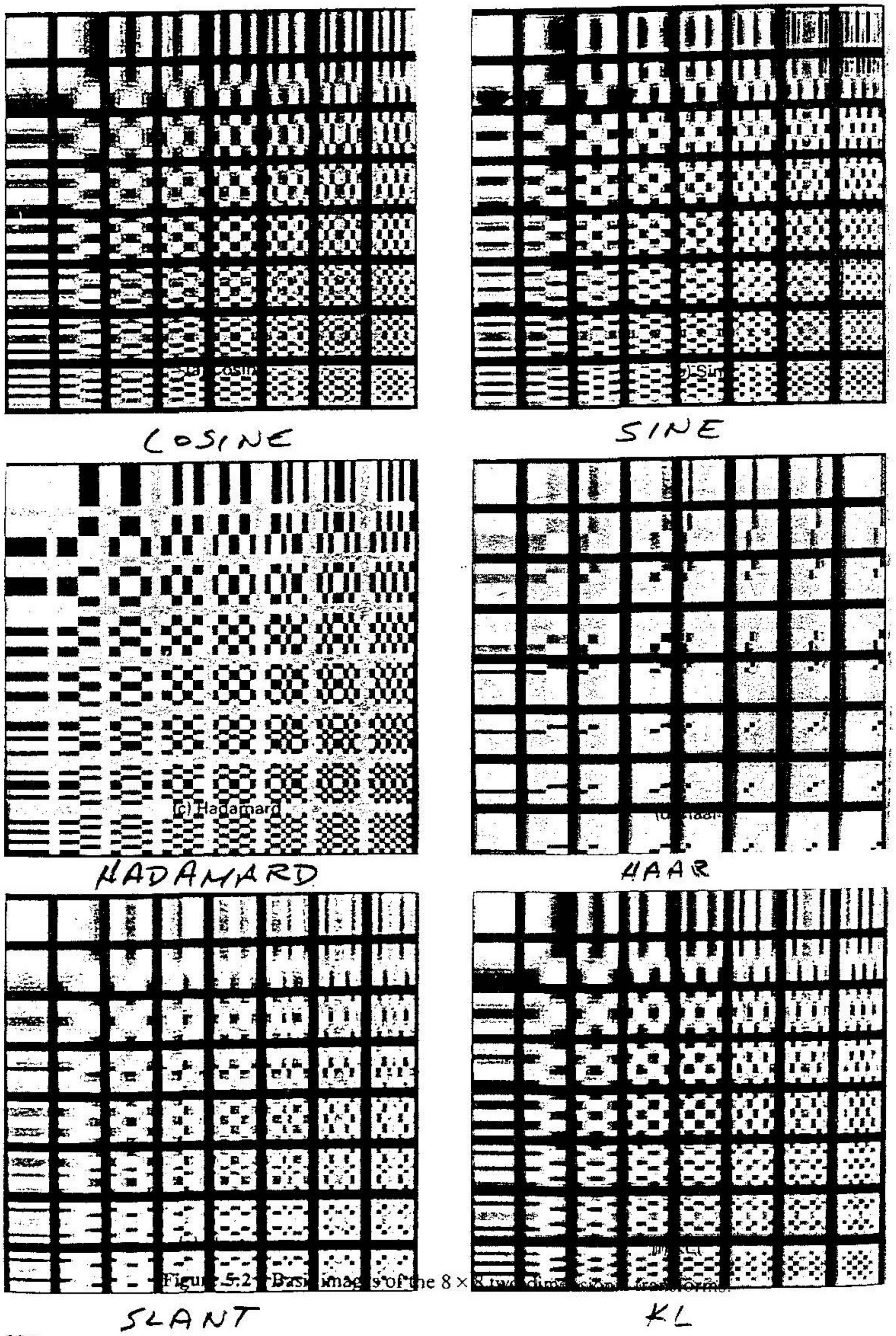


Figure 5.2 Basis images of the 8 × 8 two-dimensional transforms.

8×8 2D

(a) Cosine; (b) sine; (c) unitary DFT; (d) Hadamard; (e) Haar; (f) Slant.

Figure 5.22 Basis restriction zonal filtering using different transforms with 4 : 1 sample reduction.

Again we find the cosine transform to have the best performance.

(a) Original;

(b) 4 : 1 sample reduction;

(c) 8 : 1 sample reduction;

(d) 16 : 1 sample reduction.

Sine Transform

$$v_k = \sqrt{\frac{2}{N+1}} \sum_{n=0}^{N-1} u_n \sin\left[\frac{(n+1)(k+1)\pi}{N+1}\right] k = 0,1,\ldots N-1$$

and

$$u_n = \sqrt{\frac{2}{N+1}} \sum_{k=0}^{N-1} v_k \sin\left[\frac{(n+1)(k+1)\pi}{N+1}\right] k = 0,1,\ldots N-1$$

To evaluate this fast we relate it to a FFT

$$v_k = \mathrm{Re}\left\{\alpha_k e^{-\frac{\pi i k}{2N}} \sum_{n=0}^{N-1} \tilde{u}_n e^{-\frac{2\pi i k n}{N}}\right\} = \mathrm{Re}\left\{\alpha_k w_{2N}^k DFT(u)\right\}$$

where

$$\tilde{u}_n = u_{2n}$$

$$\tilde{u}_{N-n-1} = u_{2n} + 1 \quad 0 \le n \le \frac{N}{2} - 1$$

Inverse

```
%

% Y = DCT(X) returns the discrete cosine transform of X. The
% vector Y is the same size as X and contains the discrete
% cosine transform coefficients.
%
% Author(s): C. Thompson, 2-12-93
%            S. Eddins, 10-26-94, revised
% Copyright 1993-1998 The MathWorks, Inc.  All Rights Reserved.
% $Revision: 5.3 $  $Date: 1997/11/24 16:21:02 $

% References:
%    1) A. K. Jain, "Fundamentals of Digital Image
%       Processing", pp. 150-153.
%    2) Wallace, "The JPEG Still Picture Compression Standard",
%       Communications of the ACM, April 1991.

if rem(n,2)==1 | ~isreal(a), % odd case
% Form intermediate even-symmetric matrix.

else % even case

% Re-order the elements of the columns of x
y = [ aa(1:2:n,:); aa(n:-2:2,:) ];

% Compute weights to multiply DFT coefficients
ww = 2*exp(-i*(0:n-1)'*pi/(2*n))/sqrt(2*n);
ww(1) = ww(1) / sqrt(2);
W = ww(:,ones(1,m));

% Compute DCT using equation (5.92) in Jain
b = W .* fft(y);
end

if isreal(a), b = real(b); end
if do_trans, b = b.'; end
```

Notes


```
%

% X = IDCT(Y) inverts the DCT transform, returning the original
% vector if Y was obtained using Y = DCT(X).
%
% Author(s): C. Thompson, 2-12-93
%            S. Eddins, 10-26-94, revised
% Copyright 1993-1998 The MathWorks, Inc. All Rights Reserved.
% $Revision: 5.3 $ $Date: 1997/11/24 16:21:02 $

% References:
%     1) A. K. Jain, "Fundamentals of Digital Image
%        Processing", pp. 150-153.
%     2) Wallace, "The JPEG Still Picture Compression Standard",
%        Communications of the ACM, April 1991.

if rem(n,2)==1 | ~isreal(b), % odd case
% Form intermediate even-symmetric matrix.

else % even case
% Compute precorrection factor
ww = sqrt(2*n) * exp(j*pi*(0:n-1)/(2*n)).';
ww(1) = ww(1)/sqrt(2);
W = ww(:,ones(1,m));

% Compute x tilde using equation (5.93) in Jain
y = ifft(W.*bb);

% Re-order elements of each column according to equations (5.93) and
% (5.94) in Jain
a = zeros(n,m);
a(1:2:n,:) = y(1:n/2,:);
a(2:2:n,:) = y(n:-1:n/2+1,:);
end

if isreal(b), a = real(a); end
if do_trans, a = a.'; end
```

all symmetrizable

DCT 1 $\quad \cos \frac{jk\pi}{N-1}$

$$\begin{pmatrix} 2 & -2 & & & \\ -1 & 2 & -1 & & \\ & & \ddots & & \\ & & -1 & 2 & -1 \\ & & & -2 & 2 \end{pmatrix}$$

DCT 2 $\quad \cos \frac{(j+\frac{1}{2})k\pi}{N}$

$$\begin{pmatrix} 1 & -1 & & & \\ -1 & 2 & -1 & & \\ & & \ddots & & \\ & & -1 & 2 & -1 \\ & & & -1 & 1 \end{pmatrix}$$

DCT 3 $\quad \cos \frac{j(k+\frac{1}{2})\pi}{N}$

$$\begin{pmatrix} 2 & -2 & & & \\ -1 & 2 & -1 & & \\ & & \ddots & & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix}$$

DCT 4 $\quad \cos \frac{(j+\frac{1}{2})(k+\frac{1}{2})\pi}{N}$

$$\begin{pmatrix} 1 & -1 & & & \\ -1 & 2 & -1 & & \\ & & \ddots & & \\ & & -1 & 2 & -1 \\ & & & -1 & 3 \end{pmatrix}$$

ical Analysis of the DCT Coefficient Distributions for Images

. Lam, *Member, IEEE,* and Joseph W. Goodman, *Fellow, IEEE*

des, there have been various)CT coefficients for images. ıly on fitting the empirical ith a variety of well-known ıparing their goodness-of-fit. dominant choice balancing to the empirical data. Yet, has been no mathematical o this distribution. In this ical analysis using a doubly ıich not only provides the t also leads to insights about literature. This model also ıanges in the image statistics butions.

ısforms, Gaussian distribu- robability statistics.

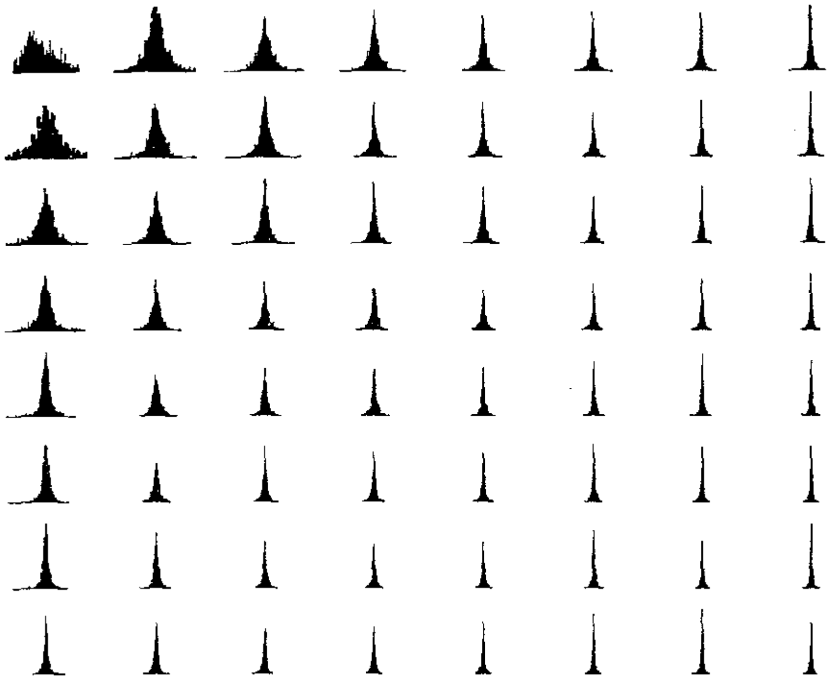


Fig. 1. Histogram of DCT coefficients of "bridge."

:ION

:t and multimedia systems, l widespread popularity for ɔntinuous-tone images. In vided into nonoverlapping h block is then subjected to ɔefore quantization and en- he compression algorithm, d by various researchers.

n understanding the distri- e more than 20 years ago. have performed the DCT the corresponding coeffi- ng statistical distribution? : instance, in quantizer de- ıhancement [1], [2]. Fig. 1 s of the DCT coefficients. picture shown in Fig. 2(a) library. The upper left co-

experimental results like Fig. 1 indicated that they resemble Laplacian distributions when the Kolmogorov–Smirnov goodness-of-fit test is used [4]. The probability density function of a Laplacian distribution can be written as

$$p(x) = \frac{\mu}{2} \exp\{-\mu|x|\}. \quad (1)$$

This is sometimes also referred to as the double exponential distribution. Since then, different researchers have tried a variety of fitting methods, such as χ^2, Kurtosis, and Watson tests. Many other possible distributions of the coefficients have also been proposed, including Cauchy, generalized Gaussian, and even a sum of Gaussians [5]–[9]. Using different pictures for the experiments, they often differ in opinion as to what distribution model is the most suitable, although the Laplacian distribution remains a popular choice balancing simplicity of the model and fidelity to the empirical data. Yet, none of them provided any analytic justification for their choices of distributions. In this paper, we investigate this problem in two steps: first, we derive the distri-