CHAPTER

Transforms for Image and Video Coding

5

CHAPTER OUTLINE

5.1 The principles of decorrelating transforms . . . 134
5.1.1 The basic building blocks . . . 134
5.1.2 Principal components and axis rotation . . . 135
5.2 Unitary transforms . . . 137
5.2.1 Basis functions and linear combinations . . . 137
5.2.2 Orthogonality and normalization . . . 138
5.2.3 Extension to 2-D . . . 139
5.3 Basic transforms . . . 140
5.3.1 The Haar transform . . . 140
5.3.2 The Walsh–Hadamard transform . . . 141
5.3.3 So why not use the discrete Fourier transform? . . . 144
5.4 Optimum transforms . . . 146
5.4.1 Discarding coefficients . . . 146
5.4.2 The Karhunen–Loeve transform (KLT) . . . 147
5.4.3 The KLT in practice . . . 148
5.5 Discrete cosine transform (DCT) . . . 149
5.5.1 Derivation of the DCT . . . 149
5.5.2 DCT basis functions . . . 151
5.5.3 Extension to 2-D: separability . . . 153
5.6 Quantization of DCT coefficients . . . 155
5.7 Performance comparisons . . . 160
5.7.1 DCT vs DFT revisited . . . 160
5.7.2 Comparison of transforms . . . 160
5.7.3 Rate–distortion performance of the DCT . . . 161
5.8 DCT implementation . . . 163
5.8.1 Choice of transform block size . . . 163
5.8.2 DCT complexity reduction . . . 164
5.8.3 Field vs frame encoding for interlaced sequences . . . 166
5.8.4 Integer transforms . . . 167
5.8.5 DCT demo . . . 167
5.9 JPEG . . . 167

Communicating Pictures. http://dx.doi.org/10.1016/B978-0-12-405906-1.00005-2

5.10 Summary 169
References 169

As we saw in Chapter 3, most natural scenes captured using regular sampling will exhibit high levels of inter-pixel correlation. This means that the actual entropy of the pixels in a region of an image is likely to be significantly lower than that given by the first order entropy or, in other words, that the spatial representation is redundant. If our aim is to compress the image, it is therefore likely that an alternative representation may be available that is more amenable to reducing redundancy. The aim of image transformation is to create such a representation by decorrelating the pixels in an image or image region. This is achieved by mapping from the image domain to an alternative domain where the pixel energy is redistributed and concentrated into a small number of coefficients. This process is sometimes referred to as energy compaction.

As we will see, this redistribution of energy does not result in data compression as the transform itself is a lossless operation. What we can achieve, however, is a concentration of energy into a small number of high-valued coefficients. These larger coefficients are likely to have a higher psychovisual significance than their low-valued counterparts and can be coded accordingly.

This chapter firstly introduces the principles and properties of decorrelating transforms and explains their relationship to principle component analysis and eigen-analysis. We introduce the optimum Karhunen–Loeve Transform (KLT) in Section 5.4.2 and, after discussing its limitations, focus the remainder of the chapter on deriving and characterizing the discrete cosine transform (DCT). The DCT is introduced in Section 5.5 and its extension to 2-D in Section 5.5.3. Quantization of DCT coefficients is presented in Section 5.6 and performance comparisons are provided in Section 5.7. We conclude with a discussion of DCT implementation and complexity in Section 5.8 and with a brief overview of JPEG in Section 5.9.

5.1 The principles of decorrelating transforms

5.1.1 The basic building blocks

Transformation presents a convenient basis for compression and this comes about through three mechanisms:

1. It provides data decorrelation and creates a frequency-related distribution of energy allowing low energy coefficients to be discarded.
2. Retained coefficients can be quantized, using a scalar quantizer, according to their perceptual importance.
3. The sparse matrix of all remaining quantized coefficients exhibits symbol redundancy which can be exploited using variable length coding.

The components in a typical transform-based image coding system are shown in Figure 5.1. As we will explain later, for the purposes of transform coding an input

For colour versions of Figures 5.14, 5.15 and 5.16 please refer to the electronic version or the website.

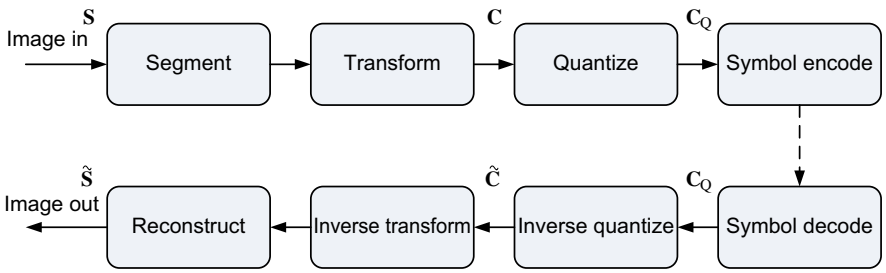

FIGURE 5.1

Typical transform-based image compression architecture.

image is normally segmented into small $N \times N$ blocks where the value of N is chosen to provide a compromise between complexity and decorrelation performance. The components shown in this figure perform the following functions:

1. **Segmentation:** This divides the image into $N \times N$ blocks—where typically $N = 8$. This is a reversible one-to-one mapping.
2. **Transformation (map or decorrelate):** This transforms the raw input data into a representation more amenable to compression. Again this is normally a reversible one-to-one mapping.
3. **Quantization:** This reduces the dynamic range of the transformed output, according to a fidelity and/or bit rate criterion, to reduce psychovisual redundancy. For correlated spatial data, the resulting block of coefficients will be sparse. This is a many-to-one mapping and is not reversible, hence once quantized, the original signal cannot be perfectly reconstructed. This is thus the basis of lossy compression.
4. **Symbol encoding (codeword assigner):** The sparsity of the quantized coefficient matrix is exploited (typically by run-length coding) to produce a string of symbols. The symbol encoder assigns a codeword (a binary string) to each symbol. The code is designed to reduce coding redundancy and it normally uses variable length codewords. This operation is reversible.

5.1.2 Principal components and axis rotation

The purpose of transformation in the context of data compression is energy compaction. This can be achieved through identification of trends in the data and then modifying the principal axes to optimally decorrelate it. It is best explained through a simple example. Consider the correlation between any two adjacent pixels in a natural image and let us represent the amplitudes of these pixels by $\{x_0, x_1\}$. A scatter plot of such adjacent pixels from a typical correlated image might look like that shown in Figure 5.2.

In Figure 5.2 (left) we observe a strong correlation characterized by the relationship $\hat{x}_1 = x_0$, as would be expected from a natural image. We can exploit this correlation by introducing a new set of axes, c_0 and c_1, as shown in Figure 5.2 which are oriented such that c_0 aligns with the principal axes of the data. Using the annotations in Figure 5.3

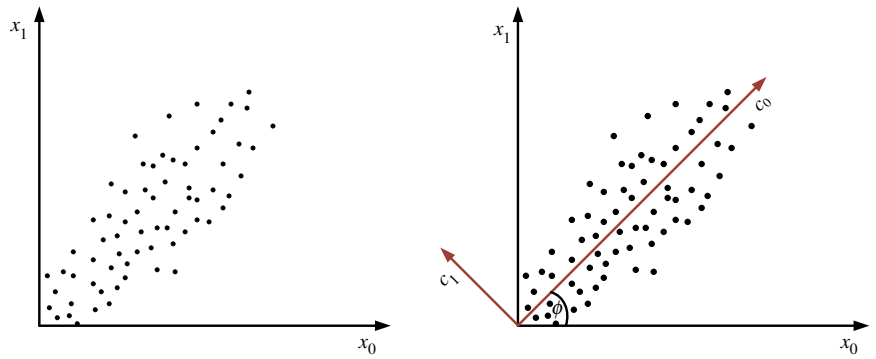

FIGURE 5.2
Plot of correlated adjacent pixel data (left) and decorrelation through rotation of principal axes (right).

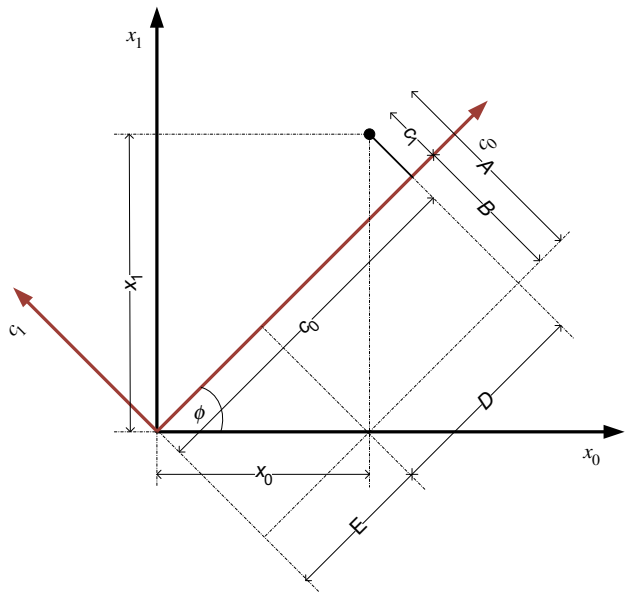

FIGURE 5.3
Relationship between original and transformed data.

we can deduce that:

$$A = x_1 \cos\phi$$
$$B = x_0 \sin\phi$$
$$D = x_1 \sin\phi$$
$$E = x_0 \cos\phi$$

Furthermore:

$$c_0 = D + E = x_0 \cos\phi + x_1 \sin\phi$$
$$c_1 = A - B = -x_0 \sin\phi + x_1 \cos\phi$$

Rearranging gives:

$$\begin{bmatrix} c_0 \\ c_1 \end{bmatrix} = \begin{bmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} \tag{5.1}$$

or in matrix vector form:

$$\mathbf{c} = \mathbf{A}\mathbf{x}$$

For correlated pixels at $\phi = 45°$:

$$\mathbf{A} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \tag{5.2}$$

As we will see in Section 5.3.2, this bears a striking resemblance to the two-point Discrete Walsh Hadamard Transform. We will also see in Section 5.4.2 that this process is equivalent to extracting the eigenvectors for this data set, i.e. $(1/\sqrt{2}, 1/\sqrt{2})(-1/\sqrt{2}, 1/\sqrt{2})$.

From equation (5.2), we can see that the row vectors for this simple transform are: $\frac{1}{\sqrt{2}}[\,1 \quad 1\,]$ and $\frac{1}{\sqrt{2}}[\,-1 \quad 1\,]$. These are effectively sum and difference operations respectively, and can be interpreted as basic low- and high-pass filters. For example, in the high-pass case:

$$H(z) = \frac{1}{\sqrt{2}}\left(1 - z^{-1}\right) \tag{5.3}$$

which has a pole at $z = 0$ and a zero at $z = 1$. This alternative view of the transform will stand us in good stead when we consider wavelets and subband filters in Chapter 6.

Let us reflect on what we have achieved in this section. Firstly, we note that the operation of rotating the axes effectively decorrelates the data, with many of the c_1 values being very small. In practice, we may set small values to zero leaving only a sparse set of coefficients which embody and preserve the main attributes of the original data set. These can then be quantized to reduce their dynamic range and coded for transmission or storage. They can then be decoded, rescaled, and inverse transformed to yield an approximation to the original data which is optimum in the mean square error sense. This is the essence of transform-based data compression as depicted in Figure 5.1.

5.2 Unitary transforms

5.2.1 Basis functions and linear combinations

We know from our introduction to Fourier analysis in Chapter 3 that we can approximate a signal using a linear combination of basis functions, such as harmonically related sines and cosines or complex exponentials. Let us assume we have N such

basis functions and indicate the contribution of each basis function $\mathbf{b}_k$ that is needed to approximate the block of data, by a value $c(k)$ where k represents the index of the basis function. Considering a one-dimensional $N \times 1$ block of data $\mathbf{x}$, we can write this relationship in matrix–vector form as follows:

$$\begin{bmatrix} x[0] \\ x[1] \\ \vdots \\ x[N-1] \end{bmatrix} = \begin{bmatrix} b_{0,0} & b_{1,0} & \dots & b_{N-1,0} \\ b_{0,1} & b_{1,1} & \dots & b_{N-1,1} \\ \vdots & \vdots & \ddots & \vdots \\ b_{0,N-1} & b_{1,N-1} & \dots & b_{N-1,N-1} \end{bmatrix} \begin{bmatrix} c(0) \\ c(1) \\ \vdots \\ c(N-1) \end{bmatrix} \tag{5.4}$$

However, if we are given an appropriate set of basis functions and some input data, then what we usually want to compute are the coefficient values or weights $c(k)$ as it is these that we will be processing, storing, and transmitting in the context of image compression. So let us rewrite equation (5.4) as follows:

$$\begin{bmatrix} c(0) \\ c(1) \\ \vdots \\ c(N-1) \end{bmatrix} = \begin{bmatrix} a_{0,0} & a_{0,1} & \dots & a_{0,N-1} \\ a_{1,0} & a_{1,1} & \dots & a_{1,N-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N-1,0} & a_{N-1,1} & \dots & a_{N-1,N-1} \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ \vdots \\ x[N-1] \end{bmatrix}$$

$$c(k) = \sum_{i=0}^{N-1} x[i]a_{k,i}; \quad k = 0 \cdots N-1$$

or

$$\mathbf{c} = \mathbf{A}\mathbf{x} \tag{5.5}$$

We have already defined the original signal in terms of our coefficients as:

$$\mathbf{x} = \mathbf{B}\mathbf{c}$$

But from equation (5.5) we can also see that the original signal can be reconstructed using the inverse transform:

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{c} \tag{5.6}$$

Hence $\mathbf{B} = \mathbf{A}^{-1}$. $\mathbf{A}$ is an $N \times N$ matrix referred to as the transform matrix and $\mathbf{c}$ is an $N \times 1$ vector of transform coefficients. The columns of the matrix $\mathbf{A}^{-1}$ are the basis functions of the transform. It can thus be observed that the signal $\mathbf{x}$ is represented by a linear combination of weighted basis functions where the weights are given by the coefficients in the column vector $\mathbf{c}$.

5.2.2 Orthogonality and normalization

Orthogonality ensures that the basis functions of a transform are independent and hence that they provide decorrelation of the input vector. Basis functions $\mathbf{a}_i$ and $\mathbf{a}_j$ are defined as orthogonal if their inner product is zero. That is:

$$\mathbf{a}_i^{\mathrm{H}}\mathbf{a}_j = 0; \quad \forall i, j \ (i \neq j) \tag{5.7}$$

Orthonormality imposes further constraints on the norm of the basis vectors in that they must have unit length, i.e. $\|\mathbf{a}_i\| = 1$. Orthonormality is useful when we wish to use the same operations for forward and inverse transformation as it preserves energy between domains, i.e.:

$$\sum_{k=0}^{N-1} c^2(k) = \sum_{i=0}^{N-1} x^2[i] \tag{5.8}$$

Furthermore, as shown above, it ensures that the forward and inverse basis function matrices are related through transposition:

$$\mathbf{a}_i^{\mathrm{H}}\mathbf{a}_j = \begin{cases} 1; & i = j \\ 0; & i \neq j \end{cases} \tag{5.9}$$

If the transform is orthonormal (or unitary) then it has a unique and computable inverse given by:

$$\mathbf{A}^{-1} = \mathbf{A}^{\mathrm{H}} \tag{5.10}$$

and hence $\mathbf{A}^{\mathrm{H}}\mathbf{A} = \mathbf{I}$. Clearly if the $\mathbf{A}$ matrix is real valued then:

$$\mathbf{A}^{-1} = \mathbf{A}^{\mathrm{T}} \tag{5.11}$$

and the inverse transform then becomes:

$$\mathbf{x} = \mathbf{A}^{\mathrm{T}}\mathbf{c} \tag{5.12}$$

In this special case, the rows of the transform matrix are the basis functions. Unitary transforms have some useful properties:

1. The autocorrelation (or covariance for non-zero mean) matrices in the signal and transform domains are related by:

$$\mathbf{R}_c = \mathbf{A}\mathbf{R}_x\mathbf{A}^{\mathrm{H}} \tag{5.13}$$

2. The total energy of the original and transformed vectors are equal (see Section 5.4.1).
3. If we select a subset of k transform coefficients, then the approximation error for a particular vector is minimized if we choose the k largest coefficients.

5.2.3 Extension to 2-D

In the case of a two-dimensional signal, let us assume that the transform is applied to an $N \times N$ block of real data $\mathbf{X}$. Assuming a separable transform, as discussed in Chapter 3 and Section 5.5.3:

$$\mathbf{C} = \mathbf{A}\mathbf{X}\mathbf{A}^{\mathrm{T}} \tag{5.14}$$

and

$$\mathbf{X} = \mathbf{A}^{\mathrm{T}}\mathbf{C}\mathbf{A} \tag{5.15}$$

Note here that equation (5.14) requires two matrix multiplications of size $N \times N$ instead of (in the non-separable case) one multiplication with a matrix of size $N^2 \times N^2$. As we will see later, the existence and exploitation of separability is important in reducing the complexity of forward and inverse transforms.

It follows that the basis functions of a 2-D transform are themselves 2-D functions, and the 2-D transform can be interpreted as expansions in terms of matrices that are obtained from the outer product of the individual 1-D basis functions, $\mathbf{a}_j$. So if the 2-D basis functions are denoted as $\boldsymbol{\alpha}_{i,j}$, and the 1-D basis functions are real-valued column vectors, then the 2-D functions are formed from the outer product of $\mathbf{a}_i$ and $\mathbf{a}_j$:

$$\boldsymbol{\alpha}_{i,j} = \mathbf{a}_i \mathbf{a}_j^{\mathrm{T}} \tag{5.16}$$

$$\boldsymbol{\alpha}_{i,j} = \begin{bmatrix} a_{i,0}a_{j,0} & a_{i,0}a_{j,1} & \cdots & a_{i,0}a_{j,N-1} \\ a_{i,1}a_{j,0} & a_{i,1}a_{j,1} & \cdots & a_{i,1}a_{j,N-1} \\ \vdots & \vdots & \ddots & \cdots \\ a_{i,N-1}a_{j,0} & a_{i,N-1}a_{j,1} & \cdots & a_{i,N-1}a_{j,N-1} \end{bmatrix} \tag{5.17}$$

Example 5.1 (Orthonormality of basis functions)
Consider the simple transform matrix from Section 5.1.2:

$$\mathbf{A} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$$

Show that this is a unitary transform.

Solution. The basis functions are orthogonal because:

$$\frac{1}{2} \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \end{bmatrix} = 0$$

and all basis functions have unit length since:

$$\|\mathbf{a}_i\| = \frac{1}{\sqrt{2}} \sqrt{(1)^2 + (1)^2} = 1$$

5.3 Basic transforms

5.3.1 The Haar transform

Consider the simple Haar transform:

$$\mathbf{H}_4 = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ \sqrt{2} & -\sqrt{2} & 0 & 0 \\ 0 & 0 & \sqrt{2} & -\sqrt{2} \end{bmatrix} \tag{5.18}$$

The first basis function creates a running sum of the input data, the second creates a difference between the first two and the second two data samples, the third creates a difference between the first two data points and similarly the basis function in the bottom row does the same for the last two data points.

Example 5.2 (Energy compaction and the Haar transform)
Consider the input vector:

$$\mathbf{x} = \begin{bmatrix} 1.0 & 0.5 & -0.5 & -1.0 \end{bmatrix}^{\mathrm{T}}$$

Show that the application of the Haar transform provides energy compaction.

Solution. A reduced number of non-zero coefficients results from the Haar transform, together with a modest amount of energy compaction, thus:

$$\mathbf{c} = \frac{1}{2}\begin{bmatrix} 0 & 3 & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}^{\mathrm{T}}$$

As we will see later, other transforms exist that can provide much greater energy compaction and decorrelation.

5.3.2 The Walsh–Hadamard transform

The Walsh and Walsh–Hadamard transforms are simple but effective ways of compressing data. They have the significant advantage that the basic transform requires no multiplications, only sums and differences. While their coding gain is lower than transforms such as the DCT which we will consider shortly, they do find application in modern video compression systems, such as H.264/AVC, where they are used in intra-frame coding.

The discrete Walsh–Hadamard transform (DWHT) is obtained from a simple rearrangement of the discrete Hadamard matrix. The Hadamard matrix is an $N \times N$ matrix with the property $\mathbf{H}\mathbf{H}^{\mathrm{T}} = N\mathbf{I}$. Higher order matrices can be found by iteratively applying the following operation:

$$\mathbf{H}_{2N} = \begin{bmatrix} \mathbf{H}_N & \mathbf{H}_N \\ \mathbf{H}_N & -\mathbf{H}_N \end{bmatrix} \tag{5.19}$$

For example:

$$\mathbf{H}_1 = 1; \quad \mathbf{H}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}; \quad \mathbf{H}_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \tag{5.20}$$

The DWHT is simply obtained from the corresponding Hadamard matrix by normalization and rearranging the rows in sequency order (i.e. in terms of the number of

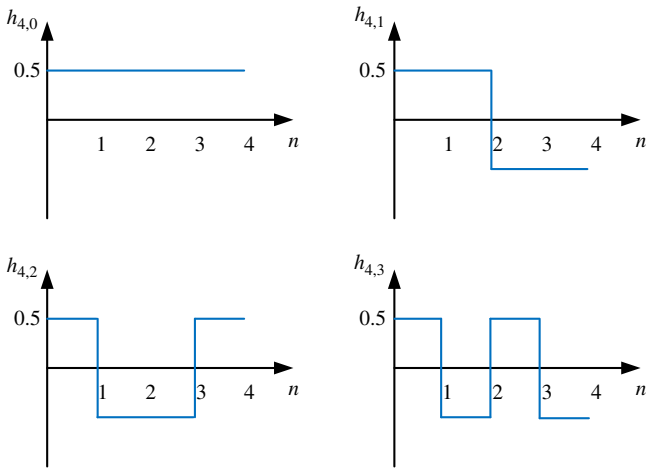

FIGURE 5.4

DWHT basis functions for $N = 4$.

sign changes). Therefore the four-point DWHT is given by:

$$\mathbf{H}_4 = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \tag{5.21}$$

The basis functions for the 1-D DWHT are depicted in Figure 5.4.

In the case of the 2-D transform, the basis functions are again formed from the outer product of the individual 1-D basis functions, $\mathbf{h}_j$. The basis functions for the 2-D 4×4 DWHT are shown in Figure 5.5.

Example 5.3 (1-D DWHT)

Compute the 1-D DWHT for the data vector $\mathbf{x} = \begin{bmatrix} 5 & 6 & 4 & 8 \end{bmatrix}^{\mathrm{T}}$:

$$\begin{bmatrix} c(0) \\ c(1) \\ c(2) \\ c(3) \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix}\begin{bmatrix} 5 \\ 6 \\ 4 \\ 8 \end{bmatrix}$$

Solution.

$$\mathbf{c} = \begin{bmatrix} 11.5 & -0.5 & 1.5 & -2.5 \end{bmatrix}^{\mathrm{T}}$$

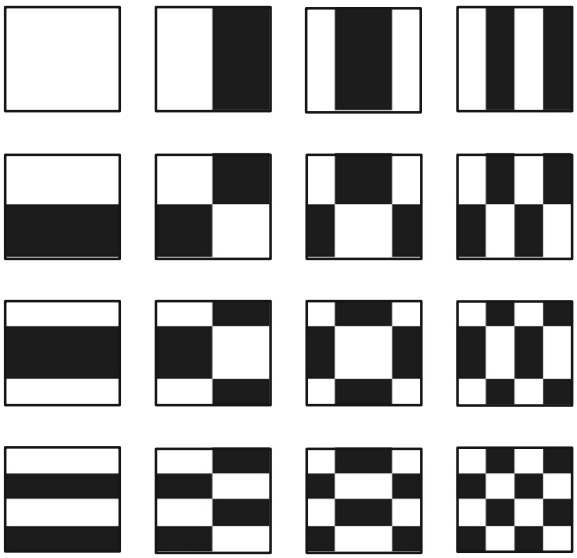

FIGURE 5.5

Basis functions for the 2-D DWHT.

Example 5.4 (2-D DWHT)

Compute the 2-D DWHT of the input image block $\mathbf{S}$ below:

$$\mathbf{S} = \begin{bmatrix} 5 & 6 & 8 & 10 \\ 6 & 6 & 5 & 7 \\ 4 & 5 & 3 & 6 \\ 8 & 7 & 5 & 5 \end{bmatrix}$$

Solution. Since the 2-D DWHT is separable, we can use equation (5.14). Letting $\mathbf{C} = \mathbf{C}'\mathbf{H}^{\mathrm{T}}$ where $\mathbf{C}' = \mathbf{HS}$:

$$\mathbf{C}' = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix}\begin{bmatrix} 5 & 6 & 8 & 10 \\ 6 & 6 & 5 & 7 \\ 4 & 5 & 3 & 6 \\ 8 & 7 & 5 & 5 \end{bmatrix} = \begin{bmatrix} 11.5 & 12 & 10.5 & 14 \\ -0.5 & 0 & 2.5 & 3 \\ 1.5 & 1 & 2.5 & 1 \\ -2.5 & -1 & 0.5 & 2 \end{bmatrix}$$

Then

$$\mathbf{C} = \begin{bmatrix} 24 & -0.5 & 1.5 & 2 \\ 2.5 & 3 & 0 & -0.5 \\ 3 & -0.5 & -0.5 & 1 \\ -0.5 & -3 & 0 & -1.5 \end{bmatrix}$$

The energy compaction and decorrelation properties of the DWHT can clearly be seen in this simple example. Recall that, in the case of the original image block, the energy was distributed fairly uniformly across the block. After transformation, the

data has been decorrelated horizontally and vertically and one dominant coefficient (top left) now contains some 93% of the energy.

Example 5.5 (2-D DWHT compression)
Taking the result of Example 5.4, set all small-valued coefficients (when $|c_{i,j}| \leq 2$) to zero and perform inverse transformation. What is the PSNR of the reconstruction if the original signal was represented with a 6 bit word length?

Solution. Now

$$\tilde{\mathbf{C}} = \begin{bmatrix} 24 & 0 & 0 & 0 \\ 2.5 & 3 & 0 & 0 \\ 3 & 0 & 0 & 0 \\ 0 & -3 & 0 & 0 \end{bmatrix}$$

and

$$\tilde{\mathbf{S}} = \mathbf{H}^{\mathrm{T}} \tilde{\mathbf{C}} \mathbf{H}$$

so

$$\tilde{\mathbf{S}} = \begin{bmatrix} 7 & 7 & 7 & 7 \\ 7 & 7 & 4 & 4 \\ 3 & 3 & 6 & 6 \\ 6 & 6 & 6 & 6 \end{bmatrix}$$

Assuming a 6 bit wordlength, the MSE = 3 and the PSNR of the reconstruction is:

$$\text{PSNR} = 10 \log_{10} \left[\frac{16(2^6 - 1)^2}{\sum_{\forall(i,j)} (s[i,j] - \tilde{s}[i,j])^2} \right] = 31.2 \text{ dB}$$

5.3.3 So why not use the discrete Fourier transform?

As a starting point for selecting a transform that decorrelates data we could do worse than consider the familiar DFT. After all, we know that the DFT is very good at representing sinusoidal data with very few coefficients. However, things are not quite that simple.

With the DFT, a windowed finite length data sequence is naturally extended by periodic extension prior to transformation. Applying the Discrete Time Fourier Series (DTFS) to this new periodic function produces the DFT coefficients. The first problem with this is that it produces a set of complex coefficients. Although this is quite useful for capturing the magnitude and phase of the underlying signal in the frequency domain, it is not particularly useful for compression since (for the case of a real input signal) we are doubling the amount of information!

Putting the complex nature of the output to one side for the moment, periodic extension can be acceptable for long stationary sequences or for shorter sequences where the extension process does not introduce discontinuities. However, for shorter

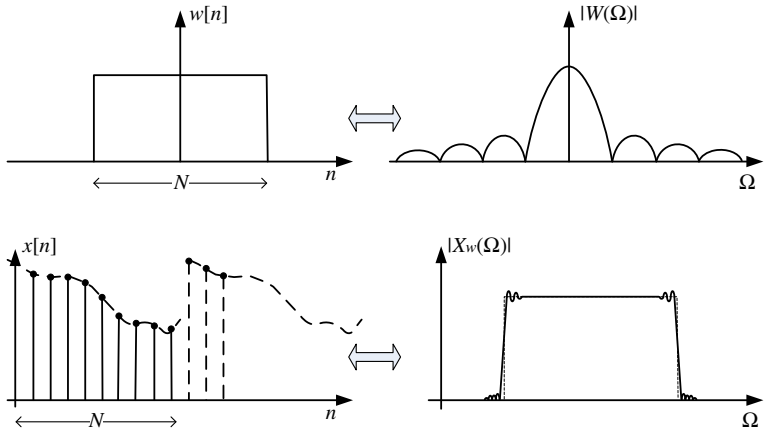

FIGURE 5.6

Spectral leakage due to windowing and periodic extension with the DFT: the rectangular window function and its magnitude spectrum (top); the input signal showing periodic extension and its spectrum (bottom).

sequences like those typical in compression, discontinuities are common and can have a major impact on performance as they produce ringing or spectral leakage in the frequency domain. This is exactly what we don't want in a compression system as it introduces more energy in the coefficient space, requiring more bits to code.

The effect of the rectangular window on the sampling and transformation processes is illustrated in Figure 5.6. Recall that multiplication of the input signal by the window function in the time (or spatial) domain is equivalent to the convolution of their spectra in the frequency domain, i.e.:

$$w[n]x[n] \Longleftrightarrow W(\Omega) * \mathcal{X}(\Omega) \tag{5.22}$$

The top two subfigures show a length N window function, alongside its spectrum, while the lower two subfigures show a sampled and windowed input signal alongside its spectrum. Since the DFT is effectively the DTFS of the periodic signal obtained by concatenating an infinite number of windowed signal samples, discontinuities exist at the window boundaries as shown in Figure 5.6. These cause spectral leakage or ripples in the frequency domain. The effects of windowing on the resulting DFT coefficients can be clearly seen.

Ringing is normally addressed in spectral analysis applications by applying a non-rectangular window function (such as a Hamming window) to the input data prior to extension. Such windows exhibit reduced frequency domain sidelobes than the conventional rectangular window, but also have a wider central lobe. Hence they reduce spectral ringing but smear sharp frequency domain edges, distorting the characteristics of the underlying signal. Again, this is exactly what we don't want in a compression system!

5.4 Optimum transforms

5.4.1 Discarding coefficients

Let us assume that we want to define a transform based on a set of basis functions that pack the most energy into the fewest transform coefficients. If we then approximate our signal by a limited number (M) of coefficients, setting the remaining coefficients to constant value, k_i, firstly we ask—What should the value of k be? Following the approach of Clarke [1]:

$$\tilde{\mathbf{x}}_M = \sum_{i=0}^{M-1} c(i)\mathbf{a}_i + \sum_{i=M}^{N-1} k_i \mathbf{a}_i$$

and the error which results from this can be computed as follows:

$$\mathbf{e}_M = \mathbf{x}_M - \tilde{\mathbf{x}}_M = \sum_{i=M}^{N-1} (c(i) - k_i)\mathbf{a}_i$$

We now wish to compute the energy in this residual signal and attempt to minimize it:

$$E\left\{e_M^2\right\} = E\left\{(c(M) - k_M)^2 \mathbf{a}_M^{\mathrm{T}}\mathbf{a}_M + (c(M) - k_M)(c(M+1) - k_{M+1})\mathbf{a}_M^{\mathrm{T}}\mathbf{a}_{M+1} + \cdots + (c(N-1) - k_{N-1})(c(N-1) - k_{N-1})\mathbf{a}_{N-1}^{\mathrm{T}}\mathbf{a}_{N-1}\right\}$$

Assuming that the transform is orthonormal (and noting that the basis functions are column vectors here):

$$\mathbf{a}_i^{\mathrm{T}}\mathbf{a}_j = \begin{cases} 1; & i = j \\ 0; & i \neq j \end{cases}$$

Hence the equation for error energy reduces to:

$$E\left[e_M^2\right] = E\left[\sum_{i=M}^{N-1} (c(i) - k_i)^2\right] \tag{5.23}$$

If we minimize this by partial differentiation with respect to k and set the result equal to 0, we can determine the optimal value of k:

$$\frac{\partial}{\partial k_i} E\left\{e_M^2\right\} = \frac{\partial}{\partial k_i} E\left\{\sum_{i=M}^{N-1} (c(i) - k_i)^2\right\} = E\left\{-2(c(i) - k_i)\right\} = 0 \text{ at minimum}$$

Hence at the minimum:

$$k_i = E\{c(i)\} = \mathbf{a}_i^{\mathrm{T}} E\{\mathbf{x}\}$$

If we assume a zero mean sequence, the solution becomes:

$$k_i = 0 \; (i = M, \ldots, N-1)$$

and hence our best approximation is:

$$\tilde{\mathbf{x}}_M = \sum_{i=0}^{M-1} c(i)\mathbf{a}_i \tag{5.24}$$

Now we can compute the basis functions which minimize the error energy in equation (5.23). So:

$$E\left\{e_M^2\right\} = E\left\{\sum_{i=M}^{N-1}(c(i))^2\right\} \tag{5.25}$$

5.4.2 The Karhunen–Loeve transform (KLT)

Also known as the Hotelling or Eigenvector transform, the basis vectors of the KLT transformation matrix, **A**, are obtained from the eigenvectors of the signal autocorrelation matrix (note the similarity to Principal Component Analysis (PCA)). Assuming the signal is derived from a random process, the resulting transform coefficients will be uncorrelated. The KLT provides the best energy compaction of any transform and yields optimal performance in the statistical (mean square error) sense for a Gaussian source, by minimizing the geometric mean of the variance of the transform coefficients. For a signal **x** (biased to remove the mean for convenience), the basis functions $\mathbf{a}_k$ can be shown to be eigenvectors of the signal autocorrelation matrix, thus:

$$\mathbf{R}_x\mathbf{a}_k = \lambda_k\mathbf{a}_k \tag{5.26}$$

where λ_k are the corresponding eigenvalues. Consider a signal $\mathbf{x} = \{x[0], x[1], \ldots, x[N-1]\}$. Assuming that the signal is a stationary random zero-mean sequence, the autocorrelation matrix for this block of pixels is given by:

$$\mathbf{R}_x = \begin{bmatrix} r_{xx}(0) & r_{xx}(1) & \cdots & r_{xx}(N-1) \\ r_{xx}(1) & r_{xx}(0) & \cdots & r_{xx}(N-2) \\ \vdots & \vdots & \ddots & \vdots \\ r_{xx}(N-1) & r_{xx}(N-2) & \cdots & r_{xx}(0) \end{bmatrix} \tag{5.27}$$

where

$$r_{xx}(k) = E\{x[n+k]x[n]\} = r_{xx}(-k)$$

As stated above, the KLT is the unitary transform that uses the eigenvectors from equation (5.26) as its basis vectors. Replacing the basis function matrix **A** in equation (5.13) with the matrix of eigenvectors, Ø, we obtain:

$$\mathbf{R}_c = \text{Ø}^{\text{H}}\mathbf{R}_x\text{Ø} \tag{5.28}$$

$$= \begin{bmatrix} \boldsymbol{\phi}_1^{\text{H}} \\ \boldsymbol{\phi}_2^{\text{H}} \\ \vdots \\ \boldsymbol{\phi}_N^{\text{H}} \end{bmatrix} \mathbf{R}_x \begin{bmatrix} \boldsymbol{\phi}_1 & \boldsymbol{\phi}_2 & \cdots & \boldsymbol{\phi}_N \end{bmatrix}$$

but from equation (5.26) this can be rewritten as:

$$\mathbf{R}_c = \begin{bmatrix} \boldsymbol{\phi}_1^{\mathrm{H}} \\ \boldsymbol{\phi}_2^{\mathrm{H}} \\ \vdots \\ \boldsymbol{\phi}_N^{\mathrm{H}} \end{bmatrix} \mathbf{R}_x \begin{bmatrix} \lambda_1\boldsymbol{\phi}_1 & \lambda_2\boldsymbol{\phi}_2 & \cdots & \lambda_N\boldsymbol{\phi}_N \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \lambda_N \end{bmatrix} \tag{5.29}$$

Hence we can see that the autocorrelation matrix is diagonalized by the KLT. It can be shown, for any autocorrelation matrix with diagonal entries, $\sigma_{c,k}^2$, that the following inequality holds:

$$\det[\mathbf{R}_c] \leq \prod_{\forall k} \sigma_{c,k}^2 \tag{5.30}$$

But, from equation (5.13) we can see that:

$$\det[\mathbf{R}_c] = \det[\mathbf{R}_x]$$

Therefore, for any unitary transform the following inequality holds:

$$\prod_{\forall k} \sigma_{c,k}^2 \geq \det[\mathbf{R}_x] \tag{5.31}$$

But, for the case of the KLT we can see that:

$$\prod_{\forall k} \sigma_{c,k}^2 = \det[\mathbf{R}_x] = \det[\mathbf{R}_c] \tag{5.32}$$

Hence the KLT minimizes the geometric mean and, as a consequence (see equation (5.43)), maximizes the coding gain for a Gaussian source. It also provides the minimum approximation error, according to equation (5.25), if a subset of coefficients is use to represent the signal.

5.4.3 The KLT in practice

The main drawback of the KLT is the dependence of the transform on the signal and thus the need to compute and transmit the transformation matrix to the decoder prior to signal reconstruction. This can be a significant overhead in the case of small block sizes and signal non-stationarity. Furthermore, for correlated data (such as an image), other transforms such as the *discrete cosine transform* (DCT) are a close approximation to the KLT. For these reasons, the DCT is the transform of choice for most image and video coding applications.

For further details on the KLT, the reader is referred to the excellent text of Clarke [1].

5.5 Discrete cosine transform (DCT)

5.5.1 Derivation of the DCT

The discrete cosine transform was first introduced by Ahmed et al. [2] and is the most widely used unitary transform for image and video coding applications. Like the discrete Fourier transform, the DCT provides information about a signal in the frequency domain. However, unlike the DFT, the DCT of a real-valued signal is itself real valued and importantly it also does not introduce artifacts due to periodic extension of the input data.

With the DFT, a finite length data sequence is naturally extended by periodic extension. Discontinuities in the time (or spatial) domain therefore produce ringing or spectral leakage in the frequency domain. This can be avoided if the data sequence is symmetrically (rather than periodically) extended prior to application of the DFT. This produces an even sequence which has the added benefit of yielding real-valued coefficients. The DCT is not as useful as the DFT for frequency domain signal analysis due to its deficiencies when representing pure sinusoidal waveforms. However, in its primary role of signal compression, it performs exceptionally well.

As we will see, the DCT has good energy compaction properties and its performance approaches that of the KLT for correlated image data. Furthermore, unlike the KLT, its basis functions are independent of the signal. The 1-D DCT, in its most popular form, is given by:

$$c(k) = \sqrt{\frac{2}{N}}\varepsilon_k \sum_{m=0}^{N-1} x[m] \cos\left(\frac{\pi k}{N}\left(m + \frac{1}{2}\right)\right) \tag{5.33}$$

and in 2-D by:

$$c(k,l) = 2\frac{\varepsilon_k \varepsilon_l}{\sqrt{NM}} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} x[m,n] \cos\left(\frac{\pi k}{N}\left(m + \frac{1}{2}\right)\right) \cos\left(\frac{\pi l}{N}\left(n + \frac{1}{2}\right)\right) \tag{5.34}$$

$$\varepsilon_k = \begin{cases} \frac{1}{\sqrt{2}} & k = 0 \\ 1 & \text{otherwise} \end{cases}$$

DCT derivation

Consider the 1-D signal shown in Figure 5.7. The lower subfigure illustrates how a 1-D signal can be symmetrically extended (or mirrored) to form a sequence of length $2N$. This has the clear advantage that no discontinuities are introduced and so, when we apply the DFT to create a frequency domain representation, ringing artifacts are absent.

All N elements of the original signal, $x[n]$, are duplicated to give a new sequence $x_1[n]$:

$$x_1[n] = \begin{cases} x[n]; & 0 \leq n \leq N-1 \\ x[2N-1-n]; & N \leq n \leq 2N-1 \end{cases}$$

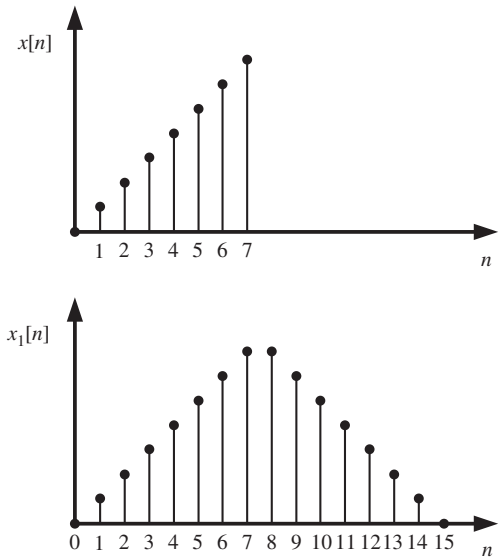

FIGURE 5.7

Symmetrical signal extension for the DCT.

The DFT of $x_1[n]$ is then given by:

$$\begin{aligned}\mathcal{X}_1(k) &= \sum_{n=0}^{N-1} x[n]W_{2N}^{nk} + \sum_{n=N}^{2N-1} x[2N-1-n]W_{2N}^{nk} \\ &= \sum_{n=0}^{N-1} x[n]\left(W_{2N}^{nk} + W_{2N}^{-(n+1)k}\right); \quad 0 \le k \le 2N-1\end{aligned}$$

Multiplying the numerator and denominator by $W_{2N}^{0.5k}$:

$$\begin{aligned}\mathcal{X}_1(k) &= W_{2N}^{-k/2} \sum_{n=0}^{N-1} x[n]\left(W_{2N}^{(n+0.5)k} + W_{2N}^{-(n+0.5)k}\right) \\ &= 2W_{2N}^{-k/2} \sum_{n=0}^{N-1} x[n]\cos\left(\frac{\pi\left(n+0.5\right)k}{N}\right)\end{aligned}$$

This is starting to look attractive as a basis for compression, apart from the rotation factor which makes the result complex and non-symmetric. Since this term contributes nothing in terms of compression performance, the DCT is normally defined without it as follows:

$$c(k) = W_{2N}^{k/2}\mathcal{X}_1(k)$$

and then normalized to give:

$$c(k) = \sqrt{\frac{2}{N}}\varepsilon_k \sum_{n=0}^{N-1} x[n] \cos\left(\frac{\pi k}{N}(n+0.5)\right) \tag{5.35}$$

where

$$\varepsilon_k = \begin{cases} \frac{1}{\sqrt{2}} & k = 0 \\ 1 & \text{otherwise} \end{cases}$$

Similarly, the inverse DCT is defined by:

$$x[n] = \sqrt{\frac{2}{N}} \sum_{k=0}^{N-1} \varepsilon_k c(k) \cos\left(\frac{\pi k}{N}(n+0.5)\right) \tag{5.36}$$

5.5.2 DCT basis functions

The basis functions are the column vectors of the inverse transform matrix. It is these that are weighted by the transform coefficients and linearly combined to form the signal approximation. However, in the case of a real-valued orthonormal transform, we have already seen that these are identical to the row vectors in the forward transform matrix.

Using equation (5.35) we can see that the DCT basis functions are given by:

$$a(k,n) = \sqrt{\frac{2}{N}}\varepsilon_k \cos\left(\frac{\pi k}{N}(n+0.5)\right); \quad 0 \le k, n \le N-1 \tag{5.37}$$

Example 5.6 (Four-point DCT basis functions)
Compute the basis functions and transform matrix for the four-point DCT.

Solution. Applying equation (5.37) we have:

$$\begin{aligned} \mathbf{A} &= \sqrt{\frac{2}{N}}\varepsilon_k \begin{bmatrix} \cos(0) & \cos(0) & \cos(0) & \cos(0) \\ \cos\left(\frac{\pi}{8}\right) & \cos\left(\frac{3\pi}{8}\right) & \cos\left(\frac{5\pi}{8}\right) & \cos\left(\frac{7\pi}{8}\right) \\ \cos\left(\frac{\pi}{4}\right) & \cos\left(\frac{3\pi}{4}\right) & \cos\left(\frac{5\pi}{4}\right) & \cos\left(\frac{7\pi}{4}\right) \\ \cos\left(\frac{3\pi}{8}\right) & \cos\left(\frac{9\pi}{8}\right) & \cos\left(\frac{15\pi}{8}\right) & \cos\left(\frac{21\pi}{8}\right) \end{bmatrix} \\ &= \sqrt{\frac{1}{2}}\varepsilon_k \begin{bmatrix} \cos(0) & \cos(0) & \cos(0) & \cos(0) \\ \cos\left(\frac{\pi}{8}\right) & \cos\left(\frac{3\pi}{8}\right) & -\cos\left(\frac{3\pi}{8}\right) & -\cos\left(\frac{\pi}{8}\right) \\ \cos\left(\frac{\pi}{4}\right) & -\cos\left(\frac{\pi}{4}\right) & -\cos\left(\frac{\pi}{4}\right) & \cos\left(\frac{\pi}{4}\right) \\ \cos\left(\frac{3\pi}{8}\right) & -\cos\left(\frac{\pi}{8}\right) & \cos\left(\frac{\pi}{8}\right) & -\cos\left(\frac{3\pi}{8}\right) \end{bmatrix} \end{aligned} \tag{5.38}$$

Returning to the four-point DCT, let us convince ourselves that this transform is orthonormal and confirm the value of ε_k. Consider, for example, the first basis

function. We can rewrite this as the vector $\mathbf{a}_0$, as follows:

$$\mathbf{a}_0 = k_0 \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}$$

Now we know that for orthonormality the basis functions must have unity norm, thus:

$$\|\mathbf{a}_0\| = k_0 \sqrt{1^2 + 1^2 + 1^2 + 1^2} = 2k_0 = 1$$

Hence:

$$k_0 = \frac{1}{2}$$

Similarly for the second basis function:

$$\|\mathbf{a}_1\| = k_1 \sqrt{\cos^2\left(\frac{\pi}{8}\right) + \cos^2\left(\frac{3\pi}{8}\right) + \cos^2\left(\frac{3\pi}{8}\right) + \cos^2\left(\frac{\pi}{8}\right)} = \sqrt{2}k_1 = 1$$

Hence:

$$k_1 = \frac{1}{\sqrt{2}}$$

And it can also be shown that:

$$k_2 = k_3 = \frac{1}{\sqrt{2}}$$

So in general we have, as expected:

$$k_i = \frac{1}{\sqrt{2}}\varepsilon_i \quad \varepsilon_i = \begin{cases} 1/\sqrt{2} & i = 0 \\ 1 & i \neq 0 \end{cases}$$

Since we now know that the DCT is orthonormal, we can simplify computation of the inverse transform since it is straightforward to show from equation (5.11) that:

$$\mathbf{A}^{-1} = \mathbf{A}^{\mathrm{T}} = \sqrt{\frac{1}{2}} \begin{bmatrix} \frac{1}{\sqrt{2}} & \cos\left(\frac{\pi}{8}\right) & \cos\left(\frac{\pi}{4}\right) & \cos\left(\frac{3\pi}{8}\right) \\ \frac{1}{\sqrt{2}} & \cos\left(\frac{3\pi}{8}\right) & -\cos\left(\frac{\pi}{4}\right) & -\cos\left(\frac{\pi}{8}\right) \\ \frac{1}{\sqrt{2}} & -\cos\left(\frac{3\pi}{8}\right) & -\cos\left(\frac{\pi}{4}\right) & \cos\left(\frac{\pi}{8}\right) \\ \frac{1}{\sqrt{2}} & -\cos\left(\frac{\pi}{8}\right) & \cos\left(\frac{\pi}{4}\right) & -\cos\left(\frac{3\pi}{8}\right) \end{bmatrix}$$

Example 5.7 (Eight-point DCT basis functions)
We can easily compute the basis functions for other length transforms. For example, compute the eight-point DCT basis functions.

Solution. The eight-point DCT basis functions can be computed using equation (5.37). These are shown in Figure 5.8.

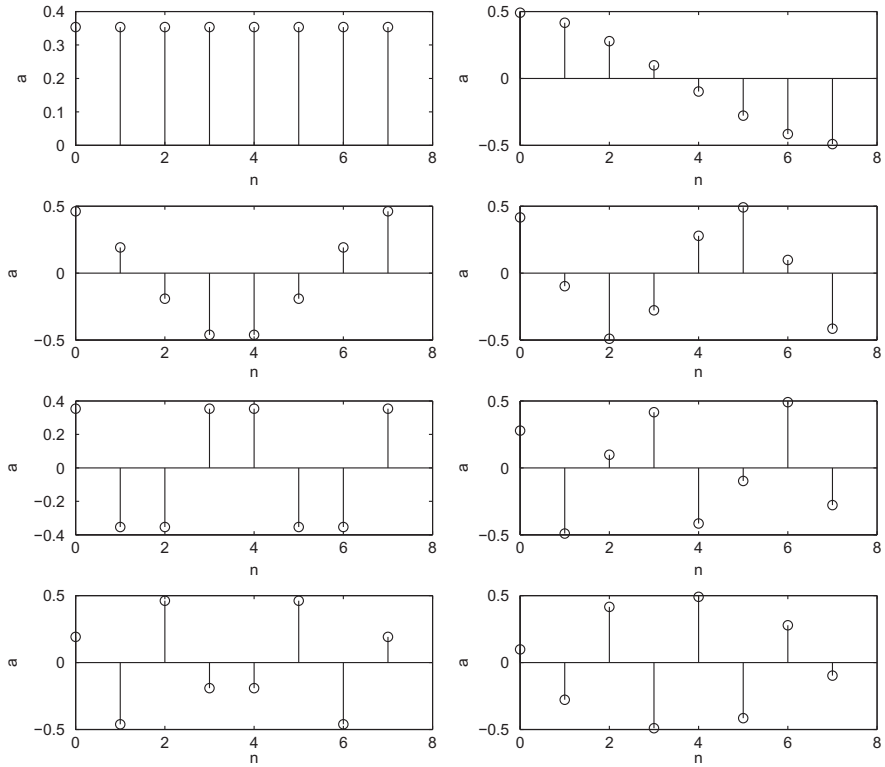

FIGURE 5.8

DCT basis functions for $N = 8$.

5.5.3 Extension to 2-D: separability

The 2-D DCT basis functions are, as described earlier, obtained from the outer product of the 1-D basis functions. These are shown for the case of the 8×8 DCT in Figure 5.9.

Let us now consider the issue of separability in the context of the 2-D DCT. Using the 8×8 transform as an example:

$$c_2(m,n) = \frac{\varepsilon_m \varepsilon_n}{4} \sum_{i=0}^{7} \sum_{j=0}^{7} s[i,j] \cos\left(\frac{(2i+1)m\pi}{16}\right) \cos\left(\frac{(2j+1)n\pi}{16}\right) \quad (5.39)$$

Nlow the 1-D DCT is given by:

$$c_1(m) = \frac{\varepsilon_m}{2} \sum_{i=0}^{7} s[i,j] \cos\left(\frac{(2i+1)\pi m}{16}\right)$$

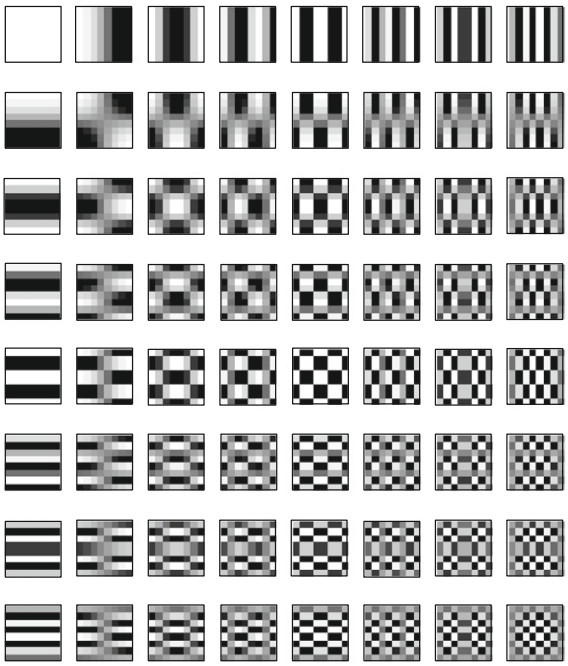

FIGURE 5.9

2-D DCT basis functions for $N = 8$.

Equation (5.39) can thus be rearranged as follows:

$$c_2(m,n) = \frac{\varepsilon_n}{2} \sum_{i=0}^{7} c_1(m) \cos\left(\frac{(2j+1)\pi n}{16}\right)$$

In other words, the 2-D DCT can be formed in two passes, firstly in the m-direction and then in the n-direction, thus:

$$c_2(m,n) = \text{DCT}_n^{\text{1-D}}\left(\text{DCT}_m^{\text{1-D}}\right) \tag{5.40}$$

Separable computation of the 2-D DCT (or indeed any separable transform) can be visualized as shown in Figure 5.10.

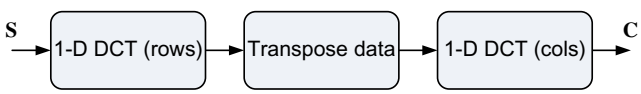

FIGURE 5.10

Separable computation of the forward DCT.

Separability has two main advantages:

1. The number of calculations is reduced—for the 8×8 case, each pass requires 8×64 multiply accumulate operations (MACs), giving a total of $2 \times 8 \times 64 = 1024$ MACs.
2. The 1-D DCT can be further decomposed to yield even greater savings (see Section 5.8.2).

Example 5.8 (2-D DCT calculation)
Calculate the DCT of the following 2-D image block:

$$\mathbf{S} = \begin{bmatrix} 21 & 19 \\ 15 & 20 \end{bmatrix}$$

Solution. The 2-D DCT transform matrix can be computed as:

$$\mathbf{A} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Recalling that:

$$\mathbf{C}_2 = \mathbf{A}\mathbf{S}\mathbf{A}^{\mathrm{T}}$$

we have

$$\begin{aligned} \mathbf{C}_2 &= \frac{1}{2} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 21 & 19 \\ 15 & 20 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \\ &= \begin{bmatrix} 37.5 & -1.5 \\ 2.5 & 3.5 \end{bmatrix} \end{aligned}$$

Now if we perform the inverse DCT on this result, we should recover the original signal:

$$\tilde{\mathbf{S}} = \frac{1}{2} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 37.5 & -1.5 \\ 2.5 & 3.5 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 21 & 19 \\ 15 & 20 \end{bmatrix} = \mathbf{S}$$

5.6 Quantization of DCT coefficients

As we have discussed previously, the process of energy compaction in itself does not provide any data compression. For example, if we have an 8×8 block of spatial domain pixel values with a dynamic range of 8 bits and we transform these to an 8×8 block of transform domain coefficients, also requiring 8 bits each, then we have gained little. What we do achieve, however, is a concentration of energy into a small number of high-valued coefficients. These larger coefficients are likely to have a higher psychovisual significance than their low-valued counterparts and can be coded accordingly.

Quantization is an important step in lossy compression; however, it is irreversible and causes signal degradation. As we saw in Chapter 3, the quantizer comprises a

set of decision levels and a set of reconstruction levels. One of the challenges is to perform quantization in such a way as to minimize its psychovisual impact.

The primary advantage of transform coding is that it provides both energy compaction and decorrelation. This means that the transform coefficients can be approximated as originating from a memoryless source (e.g. i.i.d. Laplacian) for which simple quantizers exist. Intra-frame transform coefficients are therefore normally quantized using a uniform quantizer, with the coefficients pre-weighted to reflect the frequency-dependent sensitivity of the HVS. A general expression which captures this is given in equation (5.41), where Q is the quantizer step-size, k is a constant and $\mathbf{W}$ is a coefficient-dependent weighting matrix obtained from psychovisual experiments:

$$c_Q(i,j) = \left\lfloor \frac{kc(i,j)}{Qw_{i,j}} \right\rceil \tag{5.41}$$

After transmission or storage, we must rescale the quantized transform coefficients prior to inverse transformation, thus:

$$\tilde{c}(i,j) = \frac{c_Q(i,j)Qw_{i,j}}{k} \tag{5.42}$$

First let us consider a simple example (Example 5.9), before progressing to the more realistic 8×8 case (Example 5.10).

Example 5.9 (Recalculating Example 5.8 with quantization)
Quantize the DCT result from Example 5.8 using the following quantization matrix (assume $k = 1$ and $Q = 1$):

$$\mathbf{W} = \begin{bmatrix} 4 & 8 \\ 8 & 8 \end{bmatrix}$$

Solution. Now

$$\mathbf{C}_Q = \lfloor \mathbf{C}_2 / W \rceil$$

So

$$\mathbf{C}_Q = \begin{bmatrix} 9 & 0 \\ 0 & 0 \end{bmatrix}$$

Now compute the inverse 2-D DCT after rescaling:

$$\tilde{\mathbf{S}} = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} 36 & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 18 & 18 \\ 18 & 18 \end{bmatrix} \neq \mathbf{S}$$

This leaves an error signal of:

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

Assuming 6 bit signals—what is the PSNR?

Example 5.10 (8×8 2-D DCT calculation with quantization)
Consider the following 8×8 image block, **S**:

$$\mathbf{S} = \begin{bmatrix} 168 & 163 & 161 & 150 & 154 & 168 & 164 & 154 \\ 171 & 154 & 161 & 150 & 157 & 171 & 150 & 164 \\ 171 & 168 & 147 & 164 & 164 & 161 & 143 & 154 \\ 164 & 171 & 154 & 161 & 157 & 157 & 147 & 132 \\ 161 & 161 & 157 & 154 & 143 & 161 & 154 & 132 \\ 164 & 161 & 161 & 154 & 150 & 157 & 154 & 140 \\ 161 & 168 & 157 & 154 & 161 & 140 & 140 & 132 \\ 154 & 161 & 157 & 150 & 140 & 132 & 136 & 128 \end{bmatrix}$$

First of all calculate the 8×8 block of transform coefficients, **C**, then quantize the resulting DCT coefficients using the quantization matrix, **W** (this is actually the default JPEG quantization matrix). Assuming $k = 1$ and $Q = 1$:

$$\mathbf{W} = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

Finally compute the rescaled DCT coefficient matrix and perform the inverse DCT to obtain the reconstructed image block, $\tilde{\mathbf{S}}$.

Solution.

$$\mathbf{C} = \begin{bmatrix} 1239 & 50 & -3 & 20 & -10 & -1 & 0 & -6 \\ 35 & -24 & 11 & 13 & 4 & -3 & 14 & -6 \\ -6 & -3 & 8 & -9 & 2 & -3 & 5 & 10 \\ 9 & -10 & 4 & 4 & -15 & 10 & 5 & 6 \\ -12 & 5 & -1 & -2 & -15 & 9 & -6 & -2 \\ 5 & 10 & -7 & 3 & 4 & -7 & -14 & 2 \\ 2 & -2 & 3 & -1 & 1 & 3 & -3 & -4 \\ -1 & 1 & 0 & 2 & 3 & -2 & -4 & -2 \end{bmatrix}$$

Quantize the DCT coefficients using the quantization matrix, $\mathbf{W}$. Assuming $k = 1$ and $Q = 1$, we have:

$$\mathbf{C}_Q = \begin{bmatrix} 77 & 5 & 0 & 1 & 0 & 0 & 0 & 0 \\ 3 & -2 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Rescaling the DCT coefficient matrix:

$$\tilde{\mathbf{C}} = \begin{bmatrix} 1232 & 55 & 0 & 16 & 0 & 0 & 0 & 0 \\ 36 & -24 & 14 & 19 & 0 & 0 & 0 & 0 \\ 0 & 0 & 16 & 0 & 0 & 0 & 0 & 0 \\ 14 & -17 & 0 & 0 & 0 & 0 & 0 & 0 \\ -18 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

and inverse transforming gives the reconstructed image block $\tilde{\mathbf{S}}$:

$$\tilde{\mathbf{S}} = \begin{bmatrix} 173 & 162 & 150 & 149 & 158 & 164 & 164 & 160 \\ 176 & 166 & 156 & 154 & 160 & 163 & 160 & 154 \\ 173 & 165 & 158 & 156 & 160 & 157 & 150 & 143 \\ 163 & 159 & 155 & 154 & 154 & 150 & 142 & 135 \\ 158 & 157 & 156 & 157 & 155 & 151 & 143 & 138 \\ 161 & 161 & 161 & 160 & 157 & 152 & 146 & 143 \\ 163 & 163 & 161 & 156 & 149 & 143 & 139 & 137 \\ 161 & 160 & 156 & 148 & 138 & 131 & 127 & 126 \end{bmatrix}$$

As we can see, the resulting matrix after quantization is close in value to the original signal. The key thing to notice is that the matrix $\tilde{\mathbf{C}}$ is now sparse in non-zero values and, as expected (because the input data is highly correlated), the energy in the DCT coefficients is compacted in the top left corner of the matrix.

As an additional exercise, calculate the PSNR for this approximation—assuming a signal wordlength of 8 bits.

An example of the effects of coefficient quantization on reconstruction quality for a range of block types is shown in Figure 5.11. It can be observed, as expected, that more textured blocks require larger numbers of coefficients in order to create a good approximation to the original content. The best reconstruction is achieved with fewer coefficients for the case of untextured blocks, as shown for the left-hand block in the figure.

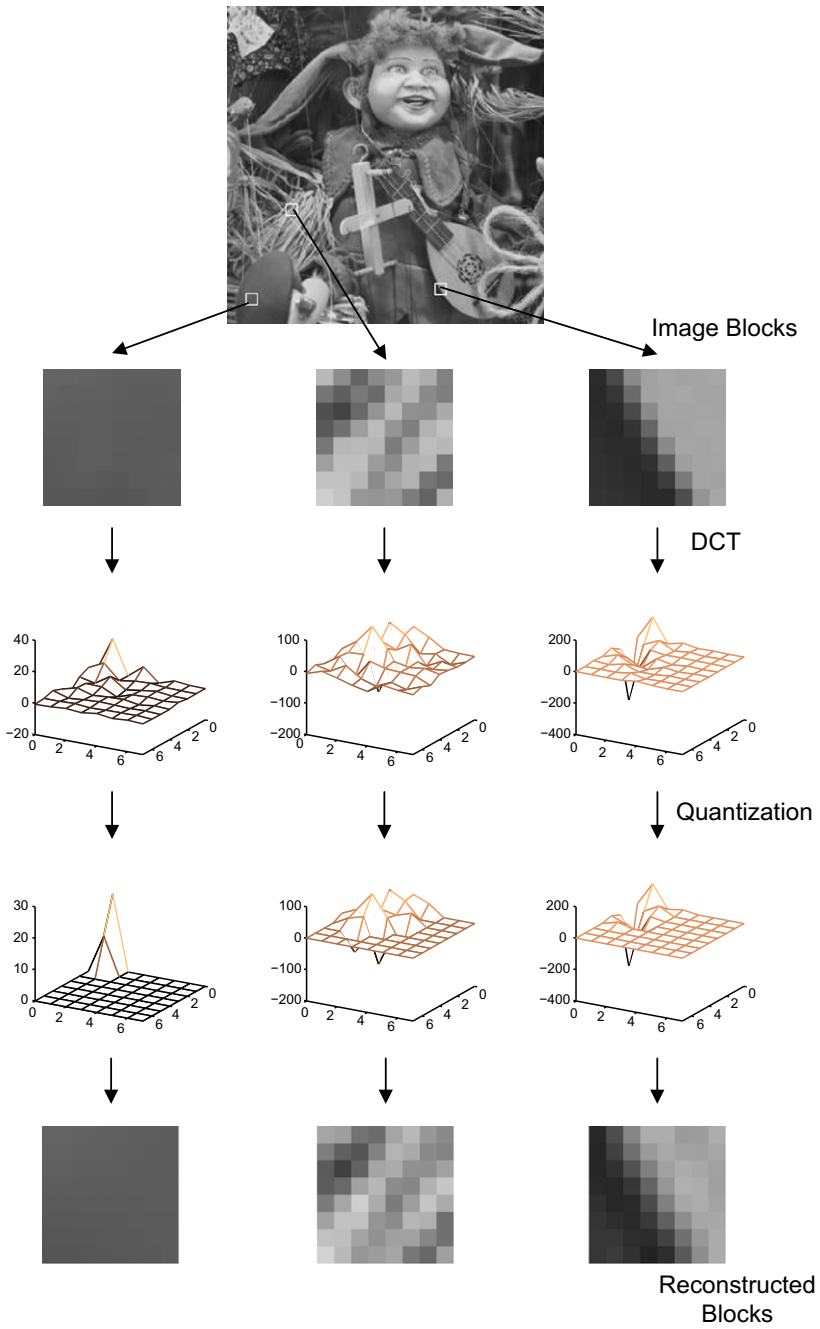

FIGURE 5.11

Effects of coefficient quantization on various types of data block.

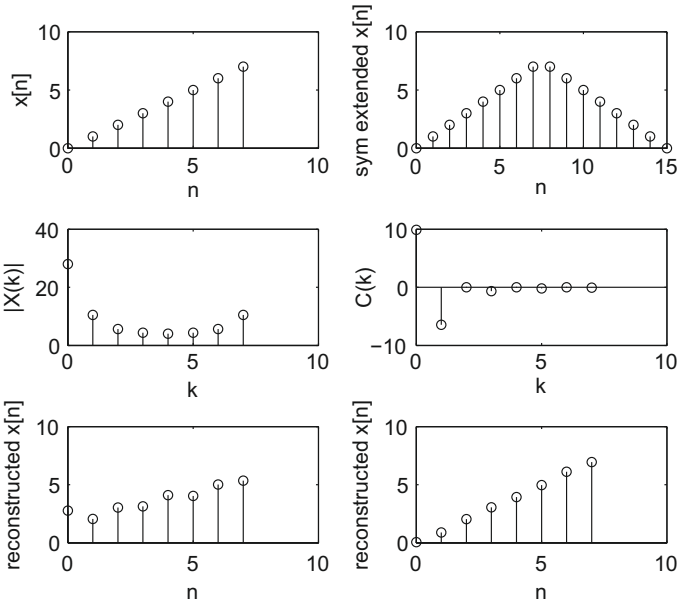

FIGURE 5.12

Comparison of DFT and DCT for compression. Top: input signal $x[n]$ and the symmetrically extended sequence $x_1[n]$. Middle: eight-point DFT$\{x[n]\}$ (magnitude only) and the eight-point DCT$\{x[n]\}$. Bottom: IDFT$\{X(k)\}$ (magnitude only) and IDCT$\{C(k)\}$.

5.7 Performance comparisons

5.7.1 DCT vs DFT revisited

Earlier we explained why the DFT does not perform well as a basis for compression. We are now in a position to compare the performance of the DCT and the DFT. Figure 5.12 shows an input signal $x[n]$ which is compressed with both an eight-point DFT and an eight-point DCT. In both cases compression is achieved by truncating the transform coefficient vector to leave only the first four coefficients, the remainder being set to zero. It can be clearly seen that the energy compaction of the DCT is superior to the DFT as is the reconstructed signal quality for a given compression strategy. Recall also that the DFT coefficients are complex and thus require approximately double the bandwidth of the DCT coefficients.

5.7.2 Comparison of transforms

Figure 5.13 shows the effects of transformation on a natural image (256×256) using the DFT, DWHT, and DCT. As expected, the DFT magnitude spectrum exhibits multiple energy concentrations, whereas the performance of the DCT provides significant energy compaction without any of the disadvantages of the DFT. The DWHT also provides good performance on this image despite its very simple structure.

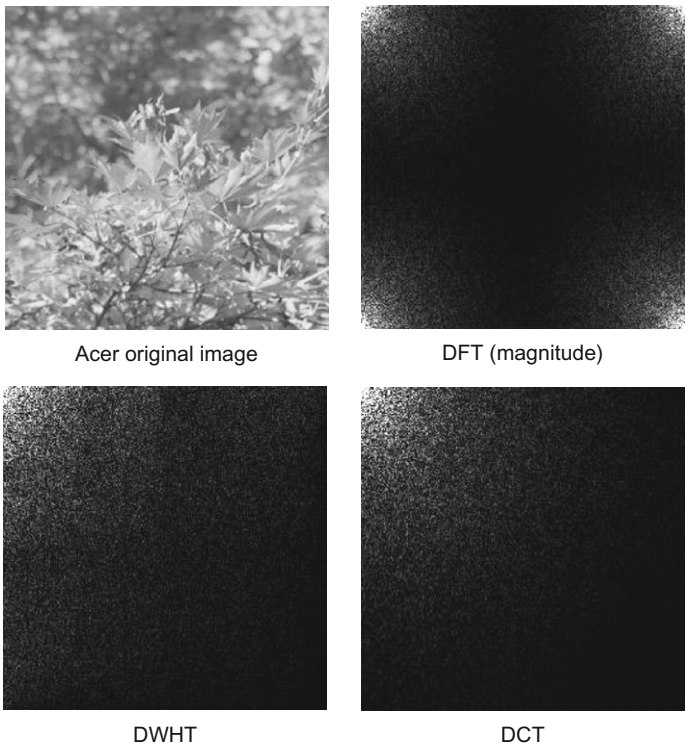

FIGURE 5.13

Comparison of KLT, DFT, DWHT, DST, and DCT on a natural image.

Computation of the KLT for a 256 × 256 image is impractical, so Figure 5.14 compares its performance with the DCT based on the use of 8 × 8 transform blocks. This figure shows two cases—the first where only nine of the 64 coefficients in each block are retained for reconstruction, and the second where 25 are used. As can be seen, the DCT comes close to the KLT in terms of quality, but importantly without the need for the transmission of basis functions. This provides additional justification of why the DCT remains so popular in image and video compression applications.

5.7.3 Rate–distortion performance of the DCT

The benefit of block-transform-based coding with the DCT is that, for correlated data, the transform compacts most of the energy in the original signal into a small number of significant coefficients. The coding gain for the block transform is defined in terms of the ratio of the arithmetic and geometric means of the transformed block variances as follows:

$$G_{TC} = 10 \log_{10} \left[\frac{\frac{1}{N} \sum_{i=0}^{N-1} \sigma_i^2}{\left(\prod_{i=0}^{N-1} \sigma_i^2 \right)^{1/N}} \right] \tag{5.43}$$

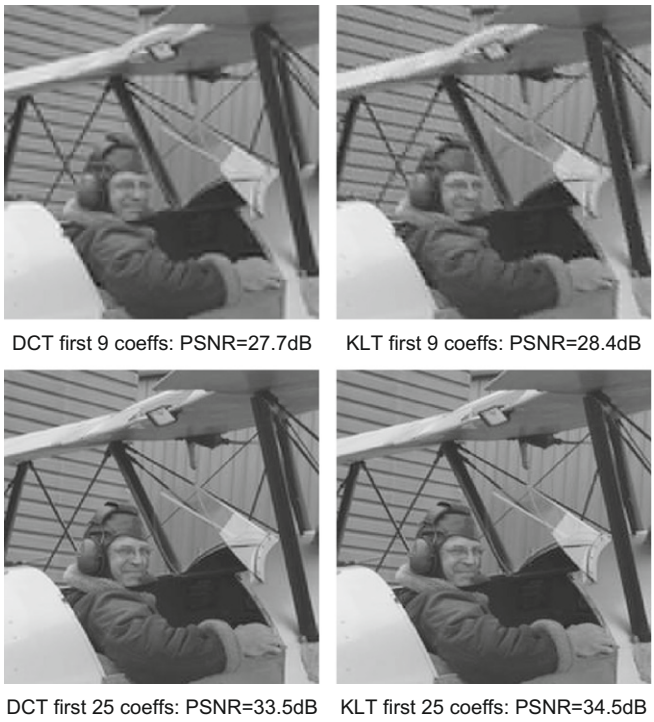

FIGURE 5.14

Comparison of KLT and DCT performance.

FIGURE 5.15

DCT performance. Left: Lena original 256 × 256. Right: compressed at 0.3 bpp.

Figure 5.15 shows an example of the performance of DCT compression (8 × 8 blocks) for a 256 × 256 resolution image at 0.3 bpp, whereas Figure 5.16 shows a comparison between the same 256 × 256 image and an equivalent 512 × 512 image. It can clearly be seen that, for the same compression ratio, the larger image is much less distorted. There are two fairly obvious reasons for this: (i) the 512 × 512 image,

FIGURE 5.16

DCT performance. Left: Lena 256 × 256 0.3 bpp. Right: Lena 512 × 512 0.3 bpp.

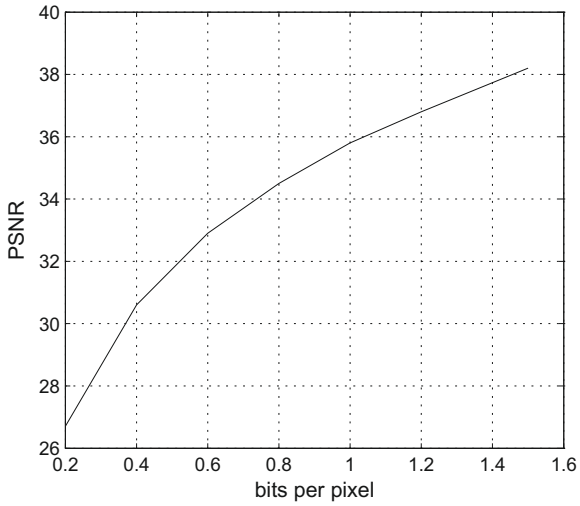

FIGURE 5.17

DCT rate–distortion performance for 256 × 256 Lena image using Huffman encoding.

despite being compressed at the same ratio, has more bits and (ii) the 512 × 512 image will exhibit higher correlation over an 8 × 8 block.

Figure 5.17 shows an example of the rate–distortion performance of the DCT coder after entropy coding (see Chapter 7) based on the Lena image.

5.8 DCT implementation

5.8.1 Choice of transform block size

What is the optimum block size for transform coding? This question is best answered by considering the data being processed and the complexity of the transform itself. We

normally seek the best compromise between decorrelation performance and complexity. As we will see below, the typical size is 8×8 pixels, although sizes down to 4×4 are used in H.264/AVC and block sizes up to 64×64 are available in H.265/HEVC. Variable block sizes are now becoming commonplace and the size is selected as part of the rate–distortion optimization (RDO) process (see Chapter 10).

It is also worth noting that the choice of transform can be data dependent. For example, the *discrete sine transform* (DST) can outperform the DCT for cases of low correlation.

DCT complexity

Using the standard 2-D DCT, the computation of each coefficient in an $N \times N$ block requires N^2 multiplications and N^2 additions. This is indicated by a complexity of order $O(N^4)$. This can be reduced to $2N$ multiplications per coefficient for the case when separability is exploited (i.e. $O(N^3)$). Hence for the case of a 4×4 block we require 8 multiplications per coefficient, for an 8×8 block we require 16 multiplications per coefficient and for a 16×16 block we require 32 multiplications per coefficient.

Taking account of both complexity and autocorrelation surface characteristics, a choice between 4×4 and 16×16 is normally used in practice.

5.8.2 DCT complexity reduction

Due to its ubiquity, many fast algorithms exist for computing the DCT. Early fast algorithms operated indirectly, exploiting the Fast Fourier Transform (FFT), with $O(N \log_2 N)$ complexity. It can be shown [1,2] that the 1-D DCT can be expressed as:

$$c(k) = \frac{2}{N}\mathbb{R}\left[e^{\frac{-jk\pi}{2N}} \sum_{n=0}^{2N-1} x[n]W^{kn}\right] \tag{5.44}$$

This type of approach is not so popular now due to the requirement for complex multiplications and additions. Other approaches take into account the unitary property of the DCT which allows it to be factorized into products of relatively sparse matrices [4]. Recursive algorithms have also been proposed [3]. The complexity of these approaches is typically $O(N^2 \log_2 N)$.

Later algorithms took direct advantage of the properties of the transform matrix, either through factorization, matrix partitioning, or by exploiting its symmetries in a divide and conquer methodology. One such approach is described below.

McGovern algorithm

In McGovern's algorithm [5], the matrix rows are decomposed, exploiting odd and even symmetries using a divide and conquer approach. The aim of this method is to

reduce the number of multiplications per coefficient:

$$\begin{bmatrix} c(0) \\ c(1) \\ c(2) \\ c(3) \\ c(4) \\ c(5) \\ c(6) \\ c(7) \end{bmatrix} = \begin{bmatrix} a_4 & a_4 & a_4 & a_4 & a_4 & a_4 & a_4 & a_4 \\ a_1 & a_3 & a_5 & a_7 & -a_7 & -a_5 & -a_3 & -a_1 \\ a_2 & a_6 & -a_6 & -a_2 & -a_2 & -a_6 & a_2 & a_2 \\ a_3 & -a_7 & -a_1 & -a_5 & a_5 & a_1 & a_7 & -a_3 \\ a_4 & -a_4 & -a_4 & a_4 & a_4 & -a_4 & -a_4 & a_4 \\ a_5 & -a_1 & a_7 & a_3 & -a_3 & -a_7 & a_1 & -a_5 \\ a_6 & -a_2 & a_2 & -a_6 & -a_6 & a_2 & -a_2 & a_6 \\ a_7 & -a_5 & a_3 & -a_1 & a_1 & -a_3 & a_5 & -a_7 \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ x[3] \\ x[4] \\ x[5] \\ x[6] \\ x[7] \end{bmatrix} \tag{5.45}$$

where:

$$a_i = \cos\left(i\frac{\pi}{16}\right)$$

$$\begin{bmatrix} c(0) \\ c(2) \\ c(4) \\ c(6) \end{bmatrix} = \begin{bmatrix} a_4 & a_4 & a_4 & a_4 \\ a_2 & a_6 & -a_6 & -a_2 \\ a_4 & -a_4 & -a_4 & a_4 \\ a_6 & -a_2 & a_2 & -a_6 \end{bmatrix} \begin{bmatrix} x[0]+x[7] \\ x[1]+x[6] \\ x[2]+x[5] \\ x[3]+x[4] \end{bmatrix} \tag{5.46}$$

$$\begin{bmatrix} c(1) \\ c(3) \\ c(5) \\ c(7) \end{bmatrix} = \begin{bmatrix} a_1 & a_3 & a_5 & a_7 \\ a_3 & -a_7 & -a_1 & -a_5 \\ a_5 & -a_1 & a_7 & a_3 \\ a_7 & -a_5 & a_3 & -a_1 \end{bmatrix} \begin{bmatrix} x[0]-x[7] \\ x[1]-x[6] \\ x[2]-x[5] \\ x[3]-x[4] \end{bmatrix} \tag{5.47}$$

The decomposition in equation (5.46) reduces the number of multiplications required from 64 ($=N^2$) in equation (5.45) to only 32 ($=2(N/2)^2$). Note that the matrix required to compute the even components of $\mathbf{x}$ is the same as that required to compute the four-point DCT. The resultant matrices can be further decomposed into a sparse form as follows:

$$\begin{bmatrix} c(0) \\ c(4) \\ c(2) \\ c(6) \\ c(7) \\ c(5) \\ c(1) \\ c(3) \end{bmatrix} = \begin{bmatrix} a_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & a_4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & a_6 & a_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & -a_2 & a_6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & a_7 & a_5 & a_1 & a_3 \\ 0 & 0 & 0 & 0 & a_5 & -a_1 & a_3 & a_7 \\ 0 & 0 & 0 & 0 & a_1 & a_3 & -a_7 & a_5 \\ 0 & 0 & 0 & 0 & a_3 & a_7 & -a_5 & -a_1 \end{bmatrix} \begin{bmatrix} g_0 \\ g_1 \\ g_2 \\ g_3 \\ g_4 \\ g_5 \\ g_6 \\ g_7 \end{bmatrix} \tag{5.48}$$

where:

$$\begin{aligned} g_0 &= x[0]+x[1]+x[2]+x[3]+x[4]+x[5]+x[6]+x[7] \\ g_1 &= x[0]+x[7]+x[3]+x[4]-x[1]-x[6]-x[2]-x[5] \\ g_2 &= x[1]+x[6]-x[2]-x[5] \\ g_3 &= x[0]+x[7]-x[3]-x[4] \end{aligned}$$

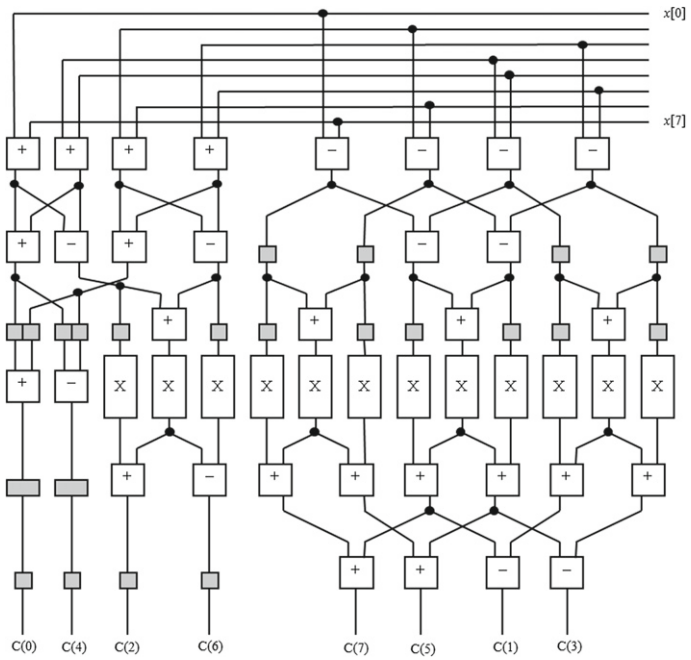

FIGURE 5.18

Efficient DCT implementation: McGovern's algorithm.

$$g_4 = x[0] - x[7]$$
$$g_5 = x[6] - x[1]$$
$$g_6 = x[3] - x[4]$$
$$g_7 = x[2] - x[5]$$

This reduces the number of multiplications required from 64 to 22. This number can be further reduced to 14 using *standard rotator products*:

$$\begin{bmatrix} x & w \\ w & -y \end{bmatrix} \begin{bmatrix} r_0 \\ r_1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} (x-w)r_0 \\ w(r_0+r_1) \\ (w+y)r_1 \end{bmatrix}$$

and still further reduced to 12 multiplications if we divide through by a_4. A bit-serial architecture for this algorithm is shown in Figure 5.18 [5] where the gray boxes represent synchronizing delay operators.

5.8.3 Field vs frame encoding for interlaced sequences

In the case of interlaced image sequences, if motion occurs between the field scans, large valued spurious DCT coefficients can occur at high spatial frequencies. These will be coarsely quantized during compression, reducing the quality of the encoded

sequence. Standards since MPEG-2 have supported the option of performing a DCT either on each field independently or on the combined frame. In the case of field pictures, all the blocks in every macroblock come from one field, whereas with an (interlaced) frame picture, frame or field DCT coding decisions can be made adaptively on a macroblock-by-macroblock basis.

When an interlaced macroblock from an interlaced frame picture is frame DCT coded, each of its four blocks has pixels from both fields. However, if the interlaced macroblock from an interlaced frame picture is field coded, each block consists of pixels from only one of the two fields.

Martin and Bull [6] produced an efficient algorithm using the primitive operator methodology of Bull and Horrocks [7]. This supports coding either one 8×8 DCT or two independent 4×4 DCT blocks. The result was compared with alternative implementations using an area–time metric and significant complexity savings were observed.

5.8.4 Integer transforms

While the DCT is in principle lossless, it can suffer from numerical mismatches between the encoder and decoder, leading to drift. This can be overcome through the use of integer transforms where integer arithmetic is employed to ensure drift-free realizations. This approach is employed in the basic H.264/AVC 4×4 integer transform which is itself derived from the 4×4 DCT. This transform is given in equation (5.49) and is discussed further in Chapter 9:

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \tag{5.49}$$

5.8.5 DCT demo

Figure 5.19 shows the graphical user interface for an informative DCT demonstration. This simple Matlab® software offers users the opportunities to visualize the effect of selecting and deselecting DCT coefficients, showing their influence on the quantized DCT matrix and on the reconstructed image. Readers can download this as part of an integrated demonstration package from http://www.bristol.ac.uk/vi-lab/demos.

5.9 JPEG

The JPEG standard, created originally in 1992 [8], is still ubiquitous. Despite the emergence of JPEG2000 as a serious contender with superior performance in 2000, JPEG has maintained its position as the dominant codec for internet and consumer use. JPEG has four primary modes:

- Baseline (sequential DCT + Huffman coding).
- Sequential lossless (predictive coding plus Huffman or arithmetic coding).

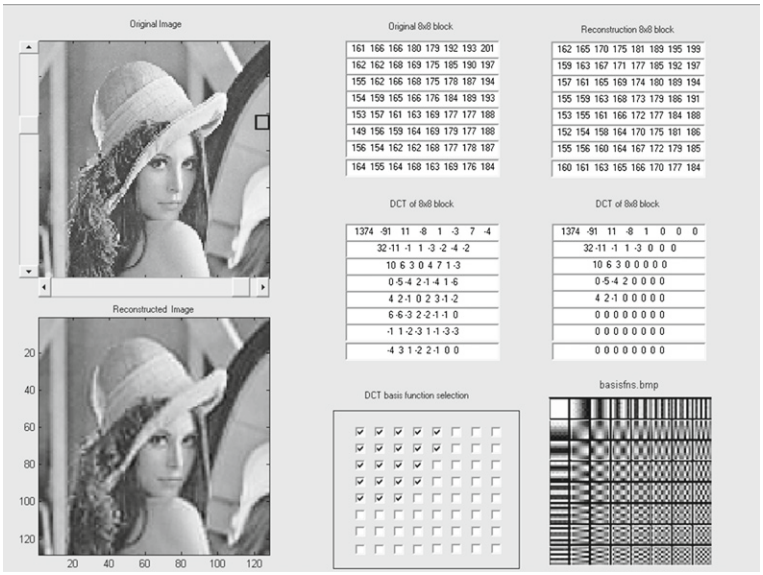

FIGURE 5.19

DCT demo GUI.

- Progressive DCT (bit plane progressive).
- Hierarchical.

Baseline JPEG is based on a simple block-based DCT with scalar quantization and Huffman coding. It is applicable to gray-scale and color images with up to 8 bits per sample and is described in Figure 5.20. We will examine the complete JPEG coding process in more detail in Chapter 7, after describing entropy coding.

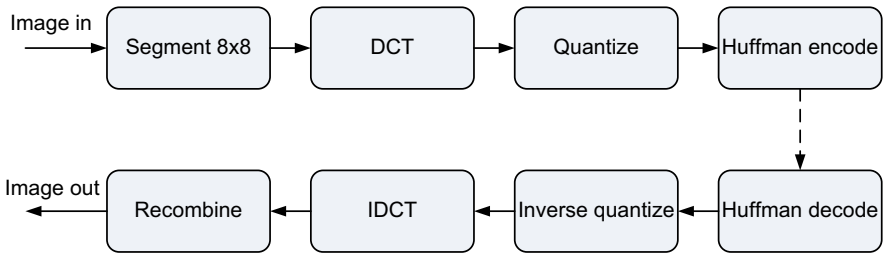

FIGURE 5.20

JPEG baseline system diagram.

5.10 Summary

In this chapter we have introduced the reader to transform coding, explaining its basic concepts and mathematical underpinnings. We have shown that the optimum transform, while providing the best energy compaction for a given data set, suffers when the data statistics are non-stationary, because of the need to continually update the transform matrix and transmit this to the decoder. Alternative transforms, such as the DCT, provide near optimum performance for correlated images with fixed bases and are the preferred choice for practical image and video compression. We have derived the DCT and demonstrated its excellent performance and its low complexity implementations.

In the next chapter we will explore another data transformation based on wavelet filterbanks and in Chapter 7 we will describe lossless coding and show how decorrelating transforms can be combined with entropy (symbol) encoding to achieve impressive compression ratios while retaining excellent perceptual image quality.

References

[1] R. Clarke, Transform Coding of Images, Academic Press, 1985.
[2] N. Ahmed, T. Natarajn, K. Rao, Discrete cosine transform, IEEE Transactions on Computers 23 (1974) 90–93.
[3] B. Lee, A new algorithm to compute the discrete cosine transform, IEEE Transactions on Acoustics, Speech, and Signal Processing 32 (1984) 1243–1245.
[4] W.-H. Chen, C. Smith, S. Fralick, A fast computational algorithm for discrete cosine transform, IEEE Transactions on Communications 25 (1977) 1004–1009.
[5] F. McGovern, R. Woods, M. Yan, Novel VLSI implementation of (8×8) point 2-D DCT, Electronics Letters 30 (8) (1994) 624–626.
[6] F. Martin, D. Bull, An improved architecture for the adaptive discrete cosine transform, in: IEEE International Symposium on Circuits and Systems, 1996, pp. 742–745.
[7] D. Bull, D. Horrocks, Primitive operator digital filters, IEE Proceedings G (Circuits, Devices and Systems) 138 (3) (1991) 401–412.
[8] ISO/IEC International Standard 10918-1, Information Technology—Digital and Coding of Continuous-Tone Still Images Requirements and Guidelines, 1992.