1 Introduction

Multimedia compression is a huge topic, which makes the task of providing a brief overview very challenging. To simplify matters, we ignore speech and audio and focus on the tools that are of interest for image and video compression. These tools can be extended to any number of dimension, as required – e.g., for the compression of multi-view video, which is currently a hot topic. Compression algorithms for such media generally consist of the following elements:

- Some form of linear transform, which helps to decorrelate the input media in some or all of its dimensions.

- Quantization in the transform domain, which introduces quantization errors (or distortion) in a controlled fashion – any compression algorithm that introduces distortion is known as a “lossy compression algorithm.” In exchange for this distortion, the transform coefficients are represented compactly using a small range of integer-valued “quantization indices.”

- Some form of variable length coding, that represents commonly occurring quantization indices using very few bits, while less common values are represented using more bits. Good transforms typically produce a lot of coefficients close to 0, so that after quantization, 0 is the most common quantization index. Coding tools endeavour to code these zeros (or equivalently, to identify the non-zero coefficients) using a very small number of bits.
Motion modelling and prediction tools which work together with linear transforms to further reduce the magnitude of the transform coefficients which must typically be quantized and coded.

In this chapter of your lecture notes, we endeavour to provide an introductory explanation of how some of these tools can be understood. The chapter focuses primarily on linear block transforms and motion compensation, using the concept of “coding gain” to provide insight into what makes a good transform or motion compensation strategy. Variable length coding and related topics are deliberately not discussed here, in the interest of brevity.

2 Linear Block Transforms

2.1 Analysis and Synthesis Matrices

In this section, we consider finite dimensional linear transforms, which map an \( n \)-dimensional input vector, \( \mathbf{x} \), into an \( m \)-dimensional output vector, \( \mathbf{y} \), according to

\[
\mathbf{y} = A^\ast \mathbf{x}, \quad \mathbf{x} = \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{m-1} \end{pmatrix}
\]

Here, \( A \) is an \( n \times m \) matrix of real or complex coefficients and \( A^\ast \) is its \( m \times n \) conjugate transpose. If the transform is real-valued then \( A^\ast = A^t \). An example of a complex-valued finite dimensional transform is the well-known DFT (Discrete Fourier Transform). We restrict our attention to invertible transforms, writing the inverse as

\[
\mathbf{x} = S \mathbf{y}
\]

where \( S \) is a \( n \times m \) left-inverse of \( A^\ast \), i.e. \( SA^\ast = I \), the \( n \times n \) identity matrix. Note that \( S \) is not necessarily unique. Henceforth, however, we shall restrict our attention to “non-expansive” transforms, having dimension \( m = n \), using \( m \) to denote this dimension and reserving the symbol, \( n \), for other purposes. In this case, \( S \) is the unique inverse of \( A^\ast \), which we may write as

\[
S^{-1} = A^\ast, \text{ or } S = A^{-\ast}
\]
Observe that the transform coefficients may be expressed as

\[ y_q = a_q^* x, \quad q = 0, 1, \ldots m - 1 \]  

(1)

where \( a_q \) is the \( q \)’th column of the \( m \times m \) matrix, \( A \). We refer to \( a_q \) as the \( q \)’th “analysis vector”, since it “analyzes” the original vector \( x \), to determine its \( q \)’th transform coefficient. Accordingly, we refer to \( A \) as the analysis matrix. Also, the inverse transform may be expressed as

\[ x = \sum_{q=0}^{m-1} y_q s_q \]  

(2)

where \( s_q \) is the \( q \)’th column of the \( m \times m \) matrix, \( S \). We refer to \( s_q \) as the \( q \)’th “synthesis vector”, since \( x \) is “synthesized” from a linear combination of the \( s_q \), with the transform coefficients serving as the weights. Accordingly, the matrix, \( S \), is known as the “synthesis matrix”.

Although it is natural to think primarily in terms of the forward transform, \( A \), the inverse transform, \( S \), often provides greater insight. In particular, the transform may be understood as a decomposition of any input vector, \( x \), as a linear combination of the synthesis vectors, \( s_q \), according to equation (2). The \( s_q \) may thus be interpreted as “prototype” vectors, with \( x \) a linear combination of these prototypes. As we shall see, transforms which are important for compression are able to represent the source accurately with only a few such prototype vectors. We also refer to the \( s_q \), as “synthesis basis” vectors because they necessarily form a basis for the linear space of all possible input vectors.

In summary, the analysis and synthesis vectors are the columns of the respective analysis and synthesis matrices, \( A \) and \( S \). The synthesis vectors form a basis of prototype signals and the transform decomposes its input as a linear combination of these prototypes.

Example 1 Consider the \( m \)-point DFT (Discrete Fourier Transform), defined by the well-known equations\(^1\)

\[ y_q = \frac{1}{\sqrt{m}} \sum_{p=0}^{m-1} x_p e^{-j \frac{2\pi}{m}pq}, \quad 0 \leq q < m \]

\[ x_p = \frac{1}{\sqrt{m}} \sum_{q=0}^{m-1} y_q e^{j \frac{2\pi}{m}pq}, \quad 0 \leq p < m \]

\(^1\)Here, we use a less common normalization convention which distributes the scaling factor of \( \frac{1}{m} \) between the forward and inverse transforms.
In this case, the analysis and synthesis vectors are identical. The elements of $a_q = s_q$ are unit spaced samples of a complex sinusoid with frequency $f_q = \frac{q}{m}$; specifically, the $p^{th}$ element is

$$a_{q,p} = s_{q,p} = \frac{1}{\sqrt{m}} e^{j2\pi f_q p}$$

Thus, the DFT decomposes its input as a linear combination of complex exponential waveforms having uniformly spaced frequencies, $f_q$.

### 2.2 Blocking of 1D and 2D Signals

In most applications, the length of the signals to be transformed is unbounded, or at least far exceeds the dimension of the transform. Consequently, the source must be processed in blocks and the selection of a finite dimensional transform implies a block transform process. Figure 1 illustrates the process for one dimensional signals. We first partition the signal, $x[k]$, into contiguous blocks of $m$ samples each, denoted $x[n]$. These blocks are then independently transformed into corresponding transform blocks, denoted $y[n]$, which form an equivalent one dimensional sequence of transform coefficients, $y[k]$.

Two dimensional signals such as images, are usually partitioned into square blocks of $m \times m$ samples each, which we may represent by $m^2$-dimensional vectors, $x[n_1, n_2] = x[n]$, as illustrated in Figure 2. The order in which the $m^2$ samples in each block appear within its vector is unimportant from a conceptual point of view. When necessary, we shall assume...
that the samples are scanned into the vector in raster-scan order. Often, however, it will be more convenient to employ a two dimensional indexing notation\(^2\) for the elements of each block vector, \(x[n]\), i.e.

\[
x[n] = \begin{pmatrix}
x_{0,0}[n] \\
x_{0,1}[n] \\
\vdots \\
x_{0,m-1}[n] \\
x_{1,0}[n] \\
\vdots \\
x_{m,m}[n]
\end{pmatrix}
\]

where

\[
x_{p_1,p_2}[n] = x[mn_1 + p_1, mn_2 + p_2], \quad 0 \leq p_1, p_2 < m
\]

and \(p_1\) and \(p_2\) denote row and column indices, respectively, of the samples in the block vector. We employ a similar notation to refer to the elements, \(y_{q_1,q_2}[n]\), of each transform vector, \(y[n] = A^\ast x[n]\). Thus, we have

\[
y_{q_1,q_2}[n] = a_{q_1,q_2}^\ast x[n]; \quad \text{and} \quad x[n] = \sum_{q_1,q_2} s_{q_1,q_2} y_{q_1,q_2}[n]
\]

\(^2\)The most natural notation here is that of tensors, where the vectors are replaced by two-dimensional tensors and the analysis and synthesis matrices are replaced by four-dimensional tensors. We choose not to burden the reader with tensor notation in this treatment.
where the analysis and synthesis vectors, \( a_{q_1,q_2} \) and \( s_{q_1,q_2} \), are the \( q_2 + mq_1 \)th columns of the \( m^2 \times m^2 \) matrices, \( A \) and \( S \), respectively.

Amongst the clutter of notation being introduced here, substantial insight may be gained into the behaviour of two dimensional block transforms by observing that they represent the input image as a linear combination of “prototype image blocks”, \( s_{q_1,q_2}[n] \). Any two prototype blocks, \( s_{q_1,q_2}[n] \) and \( s_{q_1,q_2'}[n'] \), with \( n \neq n' \), are spatially disjoint. Not surprisingly, when block transforms are used in image compression, the block structure of the prototype images can sometimes be observed in the reconstructed image. We shall investigate the structure of prototype image blocks further in Section 2.6.

2.3 Separable Transforms

An obvious way to construct a two dimensional transform, \( A \), is by “separable extension” of a one dimensional transform, \( A' \). Each analysis vector, \( a_{q_1,q_2} \), of a separable transform is formed by taking the “tensor product” of the one dimensional analysis vectors, \( a'_{q_1} \) and \( a'_{q_2} \), i.e.

\[
(a_{q_1,q_2})_{p_1,p_2} = (a'_{q_1})_{p_1} (a'_{q_2})_{p_2}, \quad 0 \leq q_1, q_2, p_1, p_2 < m
\]

Similarly, each synthesis vector, \( s_{q_1,q_2} \), is the tensor product of \( s'_{q_1} \) and \( s'_{q_2} \).

A key practical advantage of separable transforms is that they may be implemented by applying the one dimensional transform first to the rows of the image and then to its columns. To clarify this point, observe that

\[
y_{q_1,q_2} = a^\ast_{q_1,q_2} x = \sum_{p_1,p_2} (a_{q_1,q_2})^\ast_{p_1,p_2} x_{p_1,p_2}
\]

\[
= \sum_{p_1} (a'_{q_1})^\ast_{p_1} \sum_{p_2} (a'_{q_2})^\ast_{p_2} x_{p_1,p_2}
\]

\[
= a'^\ast_{q_1} y'_{q_2}
\]

Thus, we first apply the one dimensional transform independently to each row, \( p_1 \), of the image, generating an intermediate two dimensional array, \( y'_{p_1,q_2} \), each column of which is a vector, \( y'_{q_2} \), to which we apply the one dimensional transform again. The procedure is illustrated in Figure 3.

Of course, the same argument shows that we can apply the transform first to the columns and then to the rows, with identical results. Furthermore, the inverse transform may clearly be implemented in an analogous
Figure 3: Separable transform implementation by one-dimensional transformation of the rows and then the columns.

manner, starting with the rows of the coefficient array, \( y_{q_1,q_2} \), and proceeding to the columns, or vice versa. Separable transforms involve a substantial reduction in complexity. To form each coefficient of a non-separable transform, we require \( m^2 \) multiplications and additions. By contrast, only \( 2m \) multiplications and additions are required to implement a separable transform\(^3\).

**Example 2** The two dimensional \( m \times m \)-point DFT is defined by

\[
y_{q_1,q_2} = \frac{1}{m} \sum_{p_1=0}^{m-1} \sum_{p_2=0}^{m-1} x_{p_1,p_2} e^{-j \frac{2\pi}{m} (p_1 q_1 + p_2 q_2)}, \quad 0 \leq q_1, q_2 < m
\]

\[
x_{p_1,p_2} = \frac{1}{m} \sum_{q_1=0}^{m-1} \sum_{q_2=0}^{m-1} y_{q_1,q_2} e^{j \frac{2\pi}{m} (p_1 q_1 + p_2 q_2)}, \quad 0 \leq p_1, p_2 < m
\]

As for the one dimensional DFT, the analysis and synthesis vectors are identical, \( a_{q_1,q_2} = s_{q_1,q_2} \). Their elements are unit spaced samples of a two dimensional complex sinusoid with vertical and horizontal frequencies, \( f_{q_1} = \frac{q_1}{m} \) and \( f_{q_2} = \frac{q_2}{m} \), respectively. It is easily verified that this transform is the separable extension of the one dimensional DFT in Example 1.

\(^3\)Further minor simplifications are possible. For example, one of the multiplicands for the column transform may be folded into the row transform. In many cases, symmetry or richer structural properties may be exploited for further simplification.
2.4 Vector Space Perspective

Recall that the definition of inner product between two $m$-dimensional vectors, $v$ and $w$, is\footnote{Formally, this is the definition of inner-product on a finite dimensional Hilbert space over the field of complex numbers, $\mathbb{C}$.}

$$\langle v, w \rangle = \sum_{p=0}^{m-1} v_i w_i^*$$

This is the familiar “dot-product”. Thus, the forward transform (or analysis) operation of equation (1) may be written as the inner product,

$$y_q = \langle x, a_q \rangle$$

A transform is said to be orthonormal if the analysis vectors are all mutually orthogonal, having unit norm (length), i.e.

\begin{align*}
\langle a_i, a_j \rangle &= 0, \quad \forall i \neq j \quad (3) \\
\langle a_i, a_i \rangle &= \|a_i\|^2 = 1, \quad \forall i \quad (4)
\end{align*}

This means that $AA^* = I$, so that $S = A$ is a unitary matrix. Equivalently, the analysis and synthesis vectors for orthonormal transforms are identical. An orthonormal transform performs an orthonormal expansion of the input signal as the sum of its projections onto each of the basis vectors, i.e.

$$x = \sum_q y_q s_q = \sum_q \langle x, s_q \rangle \cdot s_q$$

An important property of orthonormal transforms/expansions is that they are “energy preserving”, i.e.

\begin{align*}
\sum_p |x_p|^2 &= \|x\|^2 = \langle x, x \rangle = \sum_{i,j} \langle y_i s_i, y_j s_j \rangle \\
&= \sum_{i,j} y_i y_j^* \langle s_i, s_j \rangle = \sum_q |y_q|^2 = \|y\|^2
\end{align*}

In words, the sum of the squares of the input samples (energy of the input), is identical to the sum of the squares of the transform coefficients (energy of the output).

To appreciate the significance of this property for compression, let $e_y = \hat{y} - y$ denote the error introduced into the transform coefficients by
quantization. Similarly, let \( e_x = \hat{x} - x \) denote the error introduced into the reconstructed image by the entire compression system. By linearity of the transform, \( e_x = S e_y \) and if the transform is orthonormal, \( \|e_x\|^2 = \|e_y\|^2 \). In words, the error energy in the image domain is identical to the error energy in the transform domain.

An orthogonal transform is one whose analysis vectors satisfy equation (3), but not necessarily equation (4). In this case \( AA^* = D \), a diagonal matrix, with
\[
D_{ii} = \langle a_i, a_i \rangle, \quad \text{and} \quad s_i = D_{ii}^{-1} a_i
\]
Evidently, the synthesis vectors are also orthogonal and the energy preserving property persists in the modified form
\[
\sum_p |x_p|^2 = \sum_q D_{qq}^{-2} |y_q|^2
\]

More generally, any arbitrary invertible transform has the property that \( SA^* = I \), which means that the following so-called “biorthogonality” relations must hold
\[
\langle s_i, a_j \rangle = 0, \quad \forall i \neq j
\]
\[
\langle s_i, a_i \rangle = 1, \quad \forall i
\]
The term “biorthogonal transform” may thus be applied to any invertible transform, including orthogonal transforms as a special case.

2.5 Karhunen-Loève Transform

The KLT (Karhunen-Loève Transform) is an orthonormal transform of substantial theoretical significance. Since information sources are statistical in nature, let \( x \) be the outcome of an underlying random vector, \( X \). Then the transform vector, \( y = K^*x \), is an outcome of the random vector \( Y = K^*X \).

Let \( \mu_X = E[X] \) denote the mean of the random vector, \( X \), and \( C_X \) its covariance matrix, i.e.
\[
C_X = E[(X - \mu_X)(X - \mu_X)^*] = E[XX^*] - \mu_X\mu_X^*
\]
The element at row \( i \) and column \( j \) of \( C_X \) holds the covariance of the random variables, \( X_i \) and \( X_j \), i.e.
\[
\text{cov} (X_i, Y_j) = E \left[ (X_i - \mu_{X_i}) \left( X_j - \mu_{X_j} \right)^* \right] = E \left[ X_iX_j^* \right] - \mu_{X_i}\mu_{X_j}
\]
Recall that two random variables are said to be “uncorrelated” if their covariance is zero. Accordingly, $C_X$ will be a diagonal matrix if and only if the constituent random variables, $\{X_i\}_{0 \leq i < m}$ are mutually uncorrelated.

Amongst all possible orthonormal transforms, the KLT is the unique\(^5\) transform which decorrelates its input. By this, we mean that the transform vector has a diagonal covariance matrix, $C_Y$. The relationship between $C_X$ and $C_Y$ is


Since $C_Y$ is to be diagonal and the transform is to be orthonormal, i.e. $KK^* = I$, we must have

$$KC_Y = C_XK \text{ or } k_i\sigma_i^2 = C_Xk_i, \quad 0 \leq i < m$$

where $\sigma_i^2 = \text{cov}(Y_i, Y_i)$ is the variance of the $i$'th transform coefficient, and $k_i$ is the $i$'th column of $K$, i.e. the $i$'th synthesis/analysis vector of the transform. We may conclude that the $k_i$ are the eigenvectors of the symmetric matrix, $C_X$. A well-known property of symmetric matrices is that their eigenvectors are mutually orthogonal. Thus, the KLT always exists.

A popular tool for finding the KLT matrix is the well-known SVD (Singular Value Decomposition). Specifically, the SVD of $C_X$ is

$$C_X = U\Sigma V^*$$

where $U$ and $V$ are unitary (orthonormal rows/columns) and $\Sigma$ is a diagonal matrix of singular values. Since $C_X$ is symmetric, its SVD has $U = V$; setting $K = U$, we obtain

$$C_Y = K^*C_XK = U^*U\Sigma U^*U = \Sigma$$

which is diagonal. Thus, the analysis/synthesis vectors, $k_i$, of the KLT, are the columns of the unitary matrix, $U$, from the SVD of $C_X$ and the variance

\(^5\)Actually, any orthonormal decorrelating transform must be identical to the KLT, up to a permutation (re-ordering) and potential sign-flipping of the transform coefficients.
of each transform coefficient, \( \sigma^2_{Y_i} \), is the corresponding singular value. Both the KLT and the SVD are defined so as to yield coefficient variances (singular values) in decreasing order,

\[
\sigma^2_{Y_0} \geq \sigma^2_{Y_1} \geq \cdots \geq \sigma^2_{Y_{m-1}} \tag{5}
\]

As a result of this connection, the KLT and SVD are often misunderstood as synonymous.

### 2.5.1 Significance of Decorrelating Transforms

Suppose two random variables, \( X \) and \( Y \), are statistically independent. Then, by definition, their joint PDF (probability density function) is the separable product of the marginal distributions,

\[
f_{X,Y}(x, y) = f_X(x) f_Y(y), \quad \forall x, y
\]

and hence

\[
E[XY] = \int dx \int dy \cdot xyf_{X,Y}(x, y) = \int x f_X(x) dx \int y f_Y(y) dy = \mu_X \mu_Y
\]

Thus, \( \text{cov}(X, Y) = E[XY] - \mu_X \mu_Y = 0 \) and the random variables are uncorrelated. Unfortunately, the converse is not generally true. It is common for engineers to refer to two quantities as uncorrelated, as though that were tantamount to statistical independence, although the following example demonstrates the shortcomings of such an inference.

**Example 3** Let \( X \) be a zero-mean random variable, uniformly distributed on the interval, \( \left[ -\frac{1}{2}, \frac{1}{2} \right] \), and let \( Y = |X| \). Then

\[
\text{cov}(X, Y) = E[XY] - \mu_X \mu_Y = E[XY] = \int_{-\frac{1}{2}}^{\frac{1}{2}} x |x| dx = 0
\]

So \( X \) and \( Y \) are uncorrelated, but they are certainly not independent; in fact, \( Y \) is a deterministic function of \( X \).
There is one important class of distributions for which decorrelation and statistical independence are equivalent, namely Gaussian distributions. An $m$-dimensional random vector, $\mathbf{X}$, has a Gaussian (normal) distribution if its PDF has the form

$$f_{\mathbf{X}}(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^m \det(C_{\mathbf{X}})}} e^{-\frac{1}{2}(\mathbf{x} - \mu_{\mathbf{X}})^* C_{\mathbf{X}}^{-1}(\mathbf{x} - \mu_{\mathbf{X}})}$$

When the random variables, $X_0, X_1, \ldots, X_{m-1}$, are uncorrelated, $C_{\mathbf{X}}$ is diagonal and the joint distribution becomes

$$f_{\mathbf{X}}(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^m \prod_{i=0}^{m-1} \sigma_{X_i}^2}} e^{-\frac{1}{2} \sum_{i=0}^{m-1} |x_i - \mu_{X_i}|^2}$$

$$= \prod_{i=0}^{m-1} \frac{1}{\sqrt{2\pi \sigma_{X_i}^2}} e^{-\frac{1}{2} |x_i - \mu_{X_i}|^2}$$

which is a separable product of one dimensional Gaussian distributions. Thus, jointly Gaussian random variables are statistically independent if and only if they are uncorrelated.

Another important property of Gaussian distributions is that the distribution remains Gaussian under any linear transformation. Thus, if $\mathbf{X}$ is Gaussian and $\mathbf{Y} = K^* \mathbf{X}$ is its KLT, then $\mathbf{Y}$ is a Gaussian random vector with diagonal covariance matrix, meaning that the $Y_i$ are statistically independent random variables. In this case, the KLT decomposes the source vector, $\mathbf{X}$, as a linear combination of statistically independent prototype vectors, $Y_i \mathbf{k}_i$. It turns out that the KLT is in fact the optimal block transform for compression, subject to the somewhat contrived assumption that the source follows a Gaussian distribution.

For non-Gaussian distributions, we cannot hope to decompose the source into statistically independent components by means of a linear transform. Nevertheless, since decorrelation is a necessary if not sufficient condition for statistical independence, the KLT is still an excellent choice.

### 2.5.2 Principle Components and the KLT

As with any transform, the KLT may be interpreted as decomposing the source as a linear combination of prototypes, $\mathbf{k}_i$. In the case of the KLT, these prototypes are known as the “principle components” of the source. Suppose that we are free to keep only a subset of the coefficients, with indices in $\mathcal{M}$, letting the remaining coefficients default to their mean value.
(usually zero). We synthesize the source using this reduced set of coefficients as
\[ \hat{x} = \sum_{q \in \mathcal{M}} y_q k_q + \sum_{q \notin \mathcal{M}} \mu_{Y_q} k_q \]
and the expected (mean) squared error (MSE) of this approximation is
\[ E \left[ \| X - \hat{X} \|^2 \right] = E \left[ \| Y - \hat{Y} \|^2 \right] = \sum_{q \notin \mathcal{M}} \sigma_{Y_q}^2 \]
Note that we have exploited the energy preserving property of orthonormal expansions. In view of equation (5), we should select the first \( m' = |\mathcal{M}| \) coefficients,
\[ \mathcal{M} = \{0, 1, \ldots, m' - 1\} \]
so as to minimize the MSE of the approximation. In fact, it can be shown that amongst all linear transforms, the transform for which MSE is minimum, if we keep only the first \( m' < m \) coefficients, is the KLT. This is most fortuitous, because the optimum transform does not depend upon the number of coefficients, \( m' \), which we choose to keep.

Thus, if we are free to approximate \( x \) as a multiple of only one vector (plus a constant offset), the vector which will minimize the mean square approximation error is \( k_0 \). If we are free to approximate \( x \) as a linear combination of any two vectors (plus a constant offset), the selection which will minimize the mean squared error (MSE) is the vectors, \( k_0 \) and \( k_1 \), and so on. This explains the name “principle components” for the KLT synthesis vectors, \( k_q \). This property has obvious appeal for compression, since to achieve average bit-rates of less than 1 bit per sample, there will necessarily be many transform coefficients for which no information can be communicated.

When used as a two dimensional block transform, the synthesis vectors are prototype image blocks. The first prototype image block, \( k_0 \), is the first principle component and usually corresponds to a block of constant intensity samples so that \( y_0 [n] \) is the “DC” coefficient of image block \( x [n] \) and \( \hat{x} [n] = y_0 [n] k_0 \) is a piecewise constant approximation to the original image. In general, the first few principle components usually represent smoothly varying intensity patterns, since images typically contain much more energy at low spatial frequencies than at high frequencies. Figure 4 offers evidence for this behaviour. The figure illustrates the intensity patterns associated with the first few and last principle components for blocks of size \( 8 \times 8 \), where the relevant covariances are estimated by taking averages over blocks drawn from the \( 576 \times 720 \) test image, “Goldhill”, shown in Figure 5.
Figure 4: First three and last principle components for 8 × 8 blocks taken from the image, “Goldhill”.

Figure 5: 576 × 720 test image, “Goldhill”.

14
2.6 Discrete Cosine Transform

The DCT (Discrete Cosine Transform) is a real-valued orthonormal transform, whose analysis/synthesis vectors, \( s_q \), consist of unit spaced samples of cosine functions, having frequencies, \( f_q = \frac{q}{2m} \). Specifically,

\[
s_{q,p} = c_q \cos \left( 2\pi f_q \left( p + \frac{1}{2} \right) \right) ; \quad f_q = \frac{q}{2m}
\]

where the normalization factor is selected to ensure that \( ||s_q|| = 1 \); its value is

\[
c_q = \begin{cases} 
\sqrt{\frac{n}{m}} & \text{if } q = 0 \\
\sqrt{\frac{2}{m}} & \text{if } q \neq 0 
\end{cases}
\]

The orthogonality of these vectors is easily demonstrated.

The DCT and DFT have much in common. Both are orthonormal transforms (subject to appropriate normalization) whose basis vectors are unit sampled sinusoids. In the case of the DCT, the sinusoids are real-valued and their frequencies are spaced at multiples of \( \frac{1}{2m} \), while the DFT’s sinusoids are complex-valued with frequencies separated by multiples of \( \frac{1}{m} \). We will explore the relationship between the DCT and the DFT further shortly.

The two dimensional DCT is the separable extension of the one dimensional DCT, having analysis/synthesis vectors, \( s_{q_1,q_2} \), whose elements are given by

\[
(s_{q_1,q_2})_{p_1,p_2} = c_{q_1} c_{q_2} \cos \left( 2\pi f_{q_1} \left( p_1 + \frac{1}{2} \right) \right) \cos \left( 2\pi f_{q_2} \left( p_2 + \frac{1}{2} \right) \right)
\]

These are the prototype vectors of the block-based DCT transform. Figure 6 shows the intensity patterns of a few of the lowest frequency prototype blocks for the \( 8 \times 8 \) DCT transform, along with the highest frequency prototype block, \( s_{7,7} \). Notice the similarity between these intensity patterns and those of the first few and last principle components from the \( 8 \times 8 \) KLT, appearing in Figure 4. In fact, a key attribute of the DCT is its similarity to the KLT for natural image sources.

2.6.1 Relationship between the DCT and the KLT

It can be shown that the DCT approximately diagonalizes the covariance matrix of a first-order Gauss-Markov random process for which

\[
\text{cov} \left( X_{p_1,p_2} X_{p_1',p_2'} \right) = \rho |p_1 - p_1'| + |p_2 - p_2'|
\]
Figure 6: First few and last basis (prototype) images of the $8 \times 8$ DCT.

where $\rho$ is close to 1. In fact, one of the properties of any wide-sense stationary (WSS) random process is that the Fourier coefficients are uncorrelated. That is, as $m$ becomes very large, the DFT, the DCT and other related frequency transforms all diagonalize the source covariance matrix. Thus, these transforms are asymptotically equivalent to the KLT for WSS sources, up to a reordering of the coefficients.

A suitable ordering for the DCT coefficients is the "zig-zag" scan shown in Figure 7. This order is based on the observation that the power density spectra of most images tends to decrease rapidly with increasing spatial frequency; it is employed by the JPEG image compression standard and most video compression standards.

Although most images are not well modeled as WSS random processes, the DCT has been found to be a robust approximation to the KLT for natural image sources. From a practical perspective, the DCT has numerous advantages over the KLT: it is a separable transform; highly efficient implementations exist; and there is no need (or opportunity) to adapt the transform to the statistics of the source material.

### 2.6.2 Relationship between the DCT and the DFT

It is worth exploring the relationship between the DCT and the DFT further. We restrict our attention to the one dimensional case and note that the input vector, $\mathbf{x}$, consists of a single block from an underlying source sequence. Let $\tilde{x}[n]$ be the periodic sequence with period $2m$ defined by

\[
\tilde{x}[p] = x_p, \quad 0 \leq p < m
\]

\[
\tilde{x}[n] = \tilde{x}[1-n] = \tilde{x}[2m-1-n], \quad \forall n
\]

The first half of each period holds $\mathbf{x}$, while the second half of each period contains a mirror image of $\mathbf{x}$, as illustrated in Figure 8. Since $\tilde{x}[n]$ is period,
Figure 7: Zig-zag scan to visit DCT coefficients in order of roughly decreasing variance.

Figure 8: Periodic signal whose DFT is a scaled version of the DCT of $x$. 
it may be expanded in a Fourier series as

$$\hat{x}[n] = \frac{1}{\sqrt{2m}} \sum_{q=0}^{2m-1} \hat{y}_q e^{j \frac{2\pi}{2m} nq}$$

where the Fourier coefficients, $\hat{y}_q$, are given by

$$\hat{y}_q = \frac{1}{\sqrt{2m}} \sum_{n=0}^{2m-1} \hat{x}[n] e^{-j \frac{2\pi}{2m} nq}$$

i.e. by taking the $2m$-point DFT of a single period of $\hat{x}[n]$. The above expression may be manipulated as follows:

$$\hat{y}_q = \frac{1}{\sqrt{2m}} \sum_{p=0}^{m-1} x_p e^{-j \frac{\pi}{m} q p} + \sum_{p=0}^{m-1} x_p e^{-j \frac{\pi}{m} (2m-1-p)}$$

$$= \frac{1}{\sqrt{2m}} e^{j \frac{\pi}{m} q} \sum_{p=0}^{m-1} x_p \left( e^{-j \frac{\pi}{m} (p+\frac{1}{2})} + e^{j \frac{\pi}{m} (p+\frac{1}{2})} \right)$$

$$= \sqrt{\frac{2}{m}} e^{j \frac{\pi}{m} q} \sum_{p=0}^{m-1} x_p \cos \left( 2\pi \frac{q}{2m} \left( p + \frac{1}{2} \right) \right)$$

$$= \left( \sqrt{\frac{2}{m}} e^{-j \frac{\pi}{m} q} \right) y_q$$

So the DCT coefficients of $x$ are related to the $2m$-point DFT coefficients of the symmetrically extended sequence, $\hat{x}[n]$, by a constant scale factor, which is not signal dependent, i.e.

$$y_q = \left( c_q \sqrt{\frac{m}{2}} e^{-j \frac{\pi}{2m} q} \right) \hat{y}_q$$

One of many important consequences of this connection between the DCT and the DFT is that we can exploit efficient FFT (Fast Fourier Transform) algorithms which have been developed for computing the DFT. These algorithms have complexity of order $m \log_2 m$ which represents a significant saving over direct implementation with complexity of order $m^2$.

### 3 Transforms for Compression

Some of the important properties of transforms used for image compression have been outlined in the preceding sections. At this point it is appropriate
to collect some of the key arguments which have been advanced for the use of transforms in compression. We begin in Section 3.1 with some informal arguments which are intended to provide an intuitive framework for the more formal results which follow.

3.1 Intuitive Arguments

Consider a random vector, $\mathbf{X}$, of dimension $m$ and let $A$ be the unitary $m \times m$ analysis matrix of an orthonormal transform, which maps outcomes, $\mathbf{x}$, of $\mathbf{X}$ to transform vectors, $\mathbf{y} = A^* \mathbf{x}$, which are outcomes of the random vector, $\mathbf{Y} = A^* \mathbf{X}$. We restrict our attention here to orthonormal transforms because they may be interpreted as rotation operators in the $m$-dimensional space. We shall also restrict our attention to real-valued transforms for simplicity, although the intuition extends to complex-valued transforms, such as the DFT. In the simplest case where $m = 2$, all real-valued unitary matrices are of the form

$$A^* = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}$$

up to a change of sign in one of the coordinates. We shall ignore the possibility of such sign flips, since they have no impact on the arguments which follow. In general, an $m$-dimensional real-valued unitary matrix may be factored into a product of $m^2$ one dimensional rotation operators,

$$A = \prod_{i=0}^{m-1} \prod_{j=i+1}^{m-1} A^{(i,j)}$$

where $A^{(i,j)}$ implements a rotation in the plane described by coordinates $i$ and $j$, i.e.

$$\left( A^{(i,j)} \right)^* = \begin{pmatrix} I & 0 & 0 & 0 & 0 \\ 0 & \cos \theta^{(i,j)} & 0 & -\sin \theta^{(i,j)} & 0 \\ 0 & 0 & I & 0 & 0 \\ 0 & \sin \theta^{(i,j)} & 0 & \cos \theta^{(i,j)} & 0 \\ 0 & 0 & 0 & 0 & I \end{pmatrix} \leftarrow \text{row } i \quad \leftarrow \text{row } j$$

Rotations in more than 3 dimensions are difficult, if not impossible to visualize. Fortunately, however, substantial intuition may be gained by considering the two dimensional case. Suppose that the distribution, $f_{\mathbf{X}}$, of the random vector, $\mathbf{X}$, is concentrated between the two axes, as shown in Figure 9a. In this case, a rotation of $\theta = 45^\circ$, leaves the transformed random
Figure 9: Joint distributions: (a) a two-dimensional random vector, $\mathbf{X}$; (b) its orthonormal transform, $\mathbf{Y}$; and (c) a random vector, $\mathbf{X}$, which does not benefit from linear transformation.

vector, $\mathbf{Y}$, with a distribution concentrated predominantly about the $y_0$ axis, as shown in Figure 9b. In the extreme case, we can ignore $y_1$ altogether without introducing substantial distortion and we have only to quantize and code a single quantity, rather than two quantities. Thus, the transform provides us with a simple way to exploit the statistical redundancy in the random vector, $\mathbf{X}$. Without the transform, more complex quantization and/or coding techniques would be required to exploit the statistical dependency between $X_0$ and $X_1$. More generally, we must quantize and code both quantities. However, we expect to spend very few bits coding the quantized symbols required to represent outcomes of $Y_1$, so that the bit-rate is approximately half that which we would expect from independent quantization and coding of $X_0$ and $X_1$. Of course, a more thorough analysis needs to consider the exact balance of bits spent on the two transform samples for a given combined distortion; this is the subject of Section 3.2.

A helpful notion for summarizing the effectiveness of an orthonormal transform is that of “energy compaction”. The most effective transform will concentrate the maximum amount of the energy from the source vector, $\mathbf{X}$, in a single transform sample, say $Y_0$. That is, $\sigma^2_{Y_0} \approx \sigma^2_{X_0} + \sigma^2_{X_1}$. Energy compaction is meaningful only for orthonormal transforms, or transforms which are approximately orthonormal, since these are the energy preserving transforms, with $\sigma^2_{Y_0} + \sigma^2_{Y_1} = \sigma^2_{X_0} + \sigma^2_{X_1}$. An energy compaction ratio may be defined as the ratio $\sigma^2_{Y_0}/\sigma^2_{Y_1}$.

It is easy to see that linear transforms are not always able to exploit the redundancy in the source. Consider, for example, the distribution indicated in Figure 9c. Clearly, there is substantial redundancy, since the distribution
is concentrated on a one dimensional manifold (a shell). However, no orthonormal transform is able to achieve any energy compaction. A suitable transform in this case would be a planar to polar coordinate transformation, which is highly non-linear.

With higher dimensions, \( m \), the potential for compression is also higher. We hope to be able to find orthonormal transforms which are able to rotate the source distribution into a low dimensional sub-space, so that the transform vector, \( Y \), is almost entirely described by a few of its elements. In the extreme case, all but one transform sample may be ignored and we have reduced the sample rate by a factor of \( m \) prior to quantization and coding. Although the concept of energy compaction does not immediately generalize to \( m > 2 \) dimensions, a related quantity with suitable properties is the ratio of the arithmetic mean of the variances, \( \sigma^2_{Y_i} \), to their geometric mean, i.e.

\[
\frac{\text{AM}}{\text{GM}} = \frac{\frac{1}{m} \sum_{i=0}^{m-1} \sigma^2_{Y_i}}{\sqrt{\prod_{i=0}^{m-1} \sigma^2_{Y_i}}} \tag{9}
\]

The arithmetic mean is dominated by the transform samples with the largest variance, while the geometric mean is dominated by the transform samples with the smallest variance. As we shall see in Section 3.2, this ratio has a useful interpretation as a coding gain.

### 3.1.1 Image Compression Experience

The reader may be somewhat sceptical of the extreme nature of the preceding arguments, in which we suggest that most of the transform samples might be quantized to 0. As it turns out, however, this is exactly what happens in practical image compression systems. As an example, we note that high quality reconstructed images can usually be obtained at compressed bit-rates of about 0.5 to 1.0 bpp (bits per pixel). This empirical observation holds for both the JPEG compression standard and its more recent successor, JPEG2000. Since the number of transform samples is the same as the number of image pixels and we cannot expect to spend less than 2 bits coding a non-zero transform sample value\(^6\), the majority of the samples must be quantized to zero in order to achieve such compression ratios. Experience with the baseline JPEG compression standard, for example, suggests

\(^6\)A full bit is generally required to code the sign of the non-zero sample value (most transform samples have zero mean and are largely uncorrelated with one another), while we cannot expect to spend less than one bit coding the magnitude.
that 70% to 85% of the DCT coefficients can be 0 in a high quality image representation.

3.2 Coding Gain

In this section we attempt to quantify the benefits associated with the use of transforms for image compression. There are a variety of other, more sophisticated arguments which can be used to demonstrate the value of transforms from an information theoretic point of view, but coding gain is will be sufficient for this introductory treatment.

3.2.1 Coding Gain Formulation

Coding gain expressions are a traditional means for comparing different compression techniques. The reference scheme is usually simple scalar quantization with independent coding of the source samples; for historical reasons, this is known as PCM. The distortion-rate performance of this simple direct approach can generally be modeled by

\[ D(R) = E \left[ (X' - X)^2 \right] = \sigma_X^2 g e^{-aR} \]  \hspace{1cm} (10)

Here \( X \) denotes the random variable whose outcome is the source sample being quantized and coded, \( r \geq 0 \) is the bit-rate measured in bits per sample, \( D(R) \) is the MSE achieved at rate \( R \), measured in bits per sample, \( \sigma_X^2 \) is the source variance, and \( a \) and \( g \) are constants (or more correctly, weak function of the rate, which we shall take to be a constant).

We now consider applying this same PCM scheme to the transform samples, instead of the original source samples and adjust the quantization parameters so as to achieve the same overall compressed bit-rate, \( R \), both with and without the transform. The ratio of the resulting distortions (MSE) is identified as the coding gain of the transform, i.e.

\[
\text{Coding Gain (CG)} = \frac{D^{\text{PCM}}(R)}{D^{\text{XFORM}}(R)}
\]

The key step in the procedure is the appropriate adjustment of quantization parameters. Let \( \sigma_{Y_i}^2 \) denote the variance for the \( i^{th} \) coefficient produced by a block transform. We restrict our attention to orthonormal transforms so that

\[
\sum_{i=0}^{m-1} \frac{\sigma_{Y_i}^2}{m} = \sigma_X^2
\]
Our task is to select the most appropriate operating point on the distortion-rate curve for the PCM quantizer used for each coefficient. Equivalently, we must assign rates, \( r_i \), to each coefficient, subject to
\[
\sum_{i=0}^{m-1} \frac{r_i}{m} = R
\]
Then, since the transform is orthonormal and the quantizer model of equation (10) is assumed to hold for each transform coefficient, the overall MSE will be given by
\[
D_{\text{XFORM}}(R) = \sum_{i=0}^{m-1} \frac{\sigma_{y_i}^2}{m} e^{-ar_i} \quad (11)
\]
The most appropriate rate allocation is that which minimizes the overall distortion, subject to the rate constraint. Following the method of “Lagrange multipliers”, this constrained minimization problem is equivalent to the solution to the unconstrained minimization problem
\[
\arg\min_{r_0, r_1, \ldots, r_{m-1}} \left( \sum_{i=0}^{m-1} \frac{\sigma_{y_i}^2}{m} e^{-ar_i} + \lambda \sum_{i=0}^{m-1} \frac{r_i}{m} \right)
\]
for some \( \lambda \). Setting the partial derivatives equal to zero\(^7\), in the usual way, we obtain
\[
a \sigma_{y_i}^2 e^{-ar_i} = \lambda, \quad \forall i
\]
Then
\[
D_{\text{XFORM}}(\lambda) = \frac{1}{a} \sum_{i=0}^{m-1} \frac{\lambda}{m} = \frac{\lambda}{a}
\]
and
\[
R(\lambda) = \frac{1}{a} \sum_{i=0}^{m-1} m \log_e \frac{a \sigma_{y_i}^2}{\lambda}
\]
Now, setting the compressed bit-rate for ordinary PCM (i.e. without any
\(^7\)We note that the derivatives exist only for positive rates, \( r_i \), since \( D(0) = \sigma_{y_i}^2 \) is the distortion incurred when we code nothing. The result developed here is valid only at sufficiently high rates (equiv. low distortions) such that this does not happen.
transform) equal to $R(\lambda)$, we obtain

$$D_{\text{PCM}}(\lambda) = \sigma_X^2 g e^{-aR(\lambda)}$$

$$= \sigma_X^2 g \prod_{i=0}^{m-1} \sqrt{\frac{\lambda}{a g \sigma_Y^2}}$$

$$= \frac{\sigma_X^2 \lambda}{a} \prod_{i=0}^{m-1} \frac{1}{\sigma_Y^2}$$

Finally, the coding gain becomes

$$\frac{D_{\text{PCM}}(\lambda)}{D_{\text{XFORM}}(\lambda)} = \frac{\sigma_X^2}{\prod_{i=0}^{m-1} \sigma_Y^2} = \frac{1}{m} \sum_{i=0}^{m-1} \frac{\sigma_X^2}{\sigma_Y^2}$$

(12)

The numerator of the expression is the arithmetic mean of the coefficient variances, while the denominator is a geometric mean of the variances – a value which is always greater than or equal to 1, with equality if and only if all bands have exactly the same variance, $\sigma_Y^2 = \sigma_X^2$. This is just the AM/GM ratio of equation (9). In this way, the coding gain expression confirms the intuitive arguments developed in Section 3.1.

In Section 2.5, we developed the KLT as the optimal block transform in the sense of decorrelating random vectors produced by the source. Moreover, based upon its interpretation as a decomposition of the source vectors into their principle components, we speculated that the KLT should be suitable for compression. It turns out that the KLT is also optimal in the sense that it yields the largest possible value for the coding gain expression in equation (12). It is worth restating here the observation that the DCT has similar diagonalizing properties to the KLT, when applied to images and a variety of other sources. This is of great practical value, since the KLT depends upon the source statistics, which can at best only be estimated in practical applications.

4 Introduction to JPEG

In this section, we consider only the baseline algorithm supported by the JPEG-1 compression standard. All decompression implementations are required at least to support this algorithm; in practice, few support many more of the features offered by the standard.
Figure 10: Baseline JPEG compression.

Figure 10 illustrates the central functions of the baseline compression algorithm. JPEG does not inherently impose any particular model or interpretation for colour images or any other multi-component image modalities. Instead, the image is understood as a collection of “components”, where each component is compressed essentially independently. We will revisit this topic; however, for the moment it is sufficient to appreciate that Figure 10 illustrates the compression of only a single image component.

- The first block in Figure 10 introduces an offset into the sample values so as to ensure that the data values supplied to the DCT are signed quantities whose mean value is nominally zero. Thus, for 8-bit sample values, 128 must be subtracted in this block (and added back again after decompression). For this reason, 8-bit chrominance (i.e. colour difference) components in colour images should generally be represented by values of the form $128 + c$, where $c$ is the actual zero-mean signed chrominance value. Note, however, that the concept of chrominance vs. luminance is anathema to JPEG, which does not impose any specific interpretation on the individual components which are compressed.

- The second block in the figure implements an 8x8 DCT transform.

- The DCT is an orthonormal transform. That is, the cosine basis vectors all have unit norm. Consequently, the sum of the squares (energy)

---

8 The existence of additional components affects only the scanning order in which the samples belonging to a particular component are visited.
of the image sample values is identical to the sum of the squares of the DCT coefficients. It is not hard to see that if all image samples within a single DCT block are identical, then all DCT coefficients will be zero, except for the DC coefficient which will be \( \left( \frac{8}{2\sqrt{2}} \right)^2 = 8 \) times larger than the original sample values. Thus, the nominal range of the DC coefficients generated by 8-bit image samples must be 11 bits.

- The DCT coefficients are quantized with a uniform scalar quantizer. Each DCT coefficient, \( c[k] \), is divided by a single integer in the range 1 to 255 and the result is rounded to the closest integer which is then understood as the quantized symbol value. Thus, if the dynamic range of the DC coefficients produced by 8-bit image samples is to be restored to 8 bits, the DC quantization factor must be set to 8. The quantization factor for each of the DCT coefficients is specified independently through an 8x8 quantization table, which always appears in the header of a legal JPEG compressed file. The file header supports up to four distinct quantization tables, but each component may use only one table. The association between image components and quantization tables is supplied through a special marker segment in the JPEG header.

- The JPEG standard includes two “example” quantization tables; one for luminance-type components and one which is more suitable for compressing chrominance (i.e. colour difference) components in colour imagery, but it is important to realize that these are only examples and that no legal decompressor can assume that they will be used in practice. Part III of the JPEG-1 standard specifies a mechanism for modulating the quantization table used for any given component, within the image. A scale factor may be supplied through special Huffman codewords. The scale factor is then multiplied by the nominal quantization factors for each DCT coefficient. The scale factor may be modified as often as desired\(^9\), but the additional Huffman codewords can hurt the overall compression efficiency.

- After quantization, the DC coefficient from each DCT block and the remaining 63 AC coefficients take different paths. The AC quantization symbols are first ordered into a one-dimensional sequence according to the fixed zig-zag scanning order shown in Figure 7. A run-length coding scheme is then applied which replaces each non-zero symbol in the

\(^9\) Actually, it can change only on MCU boundaries, which are defined later.
1D sequence by a triplet, \((R, C, U)\), with the following interpretation:

- \(R\) corresponds to the run-length, i.e. the number of zero-valued samples preceding the non-zero value. The maximum allowable run-length is 15. If longer runs exist, then a zero-valued category code, \(C = 0\), should be used with the maximum run-length. In this way, the value of \(R\) can always be represented with a 4-bit code.

- \(C\) is a 4-bit category code which identifies a magnitude category for the non-zero value, after which a single sign-bit, plus \(C - 1\) least significant bits of the magnitude must be sent to complete the description of the non-zero value. Thus, \(U\) always holds \(C\) bits. A degenerate value of \(C = 0\) is used if the magnitude is actually zero, which is important only when the actual run-length exceeds 15, as explained above. \(C = 1\) corresponds to a magnitude of exactly 1 so that only the sign bit need be included in \(U\). \(C = 2\) corresponds to a magnitude in the range 2 to 3, so that a single least significant bit from the magnitude must be sent. \(C = 3\) corresponds to a magnitude in the range 4 through 7 and so on. In general, category \(C\) corresponds to magnitude values in the range \(2^{C-1}\) through \(2^C - 1\).

The triplet, \((R, C, U)\), is encoded as follows. First, the 4-bit \(R\) and \(C\) values are concatenated to form an 8-bit codeword which is encoded using variable length code table, where the length of each codeword may range from 1 bit to 16 bits. The \(C\)-bit long \(U\) value is then simply appended to the variable length codeword.

A special codeword is provided to indicate that remaining coefficients in the DCT block have been quantized to zero. This, so-called “end-of-block” (EOB) symbol is identified by the otherwise useless combination, \(R = C = 0\).

- The DC coefficients are passed through a simple DPCM stage which replaces each DC coefficient value, \(c_i[0]\), with the prediction residual, \(c_i[0] - c_{i-1}[0]\), where \(i\) is the index of the DCT block within the compressor’s scanning order and the difference is taken after quantization. Each difference value produced by the DPCM stage is represented by a category/remainder pair, \((C, U)\), where the interpretation of the 4-bit category code, \(C\), and the remaining \(C\)-bit code, \(U\), are exactly the same as that described above for the AC coefficients. The 4-bit
category code is encoded using variable length code table, where the
length of each codeword must lie in the range 1 to 16 bits, while the
$C$ bits of $U$ are simply appended to this Huffman codeword.

- As with quantization tables, the JPEG file itself must always include
  the variable length code tables to be used in the algorithm described
  above. Up to four separate code tables may be supplied for coding the
  AC 8-bit run/category symbols, $(R, C)$, while another four separate
  code tables may be supplied for coding the 4-bit DPCM category codes.
  The file also provides tags to identify the particular code tables which
  are to be associated with each component.

- An often overlooked, but increasingly important requirement imposed
  by the JPEG standard is that none of the variable length code tables
  is permitted to use the entire available code space. This means that
  the codewords in each table must satisfy
  \[ \sum_{w} 2^{-|w|} < 1 \]
  where $|w|$ denotes the length (number of bits) of codeword $w$. One
  reason for this provision is that it ensures that additional codewords
  can always be added to signal other types of events. As an example,
  PART III of the JPEG standard defines an additional codeword to
  enable specification of a scale factor to be applied to the nominal
  quantization table.

- As mentioned at the beginning of this section, the JPEG standard
  does not provide explicit support for colour image compression, but
  does allow an image to be composed of multiple components, where
  the assumption is that colour images will be composed of separate
  components for each colour type (usually YCbCr, but may be RGB).
  Moreover, not all components need have the same number of samples.
  Instead, the JPEG file header identifies horizontal and vertical “sam-
  pling factors”, $H_n$ and $V_n$, with each component, $n$. The idea is that
  the image may be partitioned into small blocks, such that within each
  block component $n$ will have $H_n$ sample columns and $V_n$ sample rows.
  In the simplest case, where all components have the same number of
  samples, we have $H_n = V_n = 1$ for all components. In the case of a
  YCbCr colour image, whose chrominance components have been sub-
  sampled by two in the horizontal direction (commonly known as 4:2:2),
  the luminance component should have $H_y = 2$ and $V_y = 1$, while the
chrominance components should have $H_{cb} = H_{cr} = V_{cb} = V_{cr} = 1$. In the very common case where chrominance components are subsampled by two in both the horizontal and vertical directions, we must set $H_y = V_y = 2$, whereas the chrominance component sampling factors should all be equal to 1.

- When compressing multi-component images, the components must be partitioned into scans. All components in a scan are interleaved in a single pass through the image, but each scan must be completed before the next scan can begin. Most images have only one scan for simplicity and memory conservation, but JPEG limits the number of components which may be interleaved within a scan to at most 4.

- Within each scan, the component interleaving policy is based upon the definition of a Minimum Coded Unit (MCU). Each MCU consists of a block of $H_n$ by $V_n$ 8x8 DCT blocks from component $n$. All DCT blocks for component $n = 1$ are visited in scan-line order within the MCU, before moving to the next component, $n = 2$, and visiting all $H_2V_2$ of its DCT blocks. This continues until all DCT blocks in the MCU have been coded, after which the next MCU is considered. The order in which DCT blocks are visited is important because it affects both the order in which the relevant codewords are interleaved in the bit-stream, and the interpretation of the DPCM difference codewords used to represent the DC coefficients.

5 Introduction to Video Compression

5.1 Motion Compensated (MC) Prediction

Many popular video compression schemes are based upon a motion compensated feedback loop, as illustrated in Figure 11. The quantization block denoted $Q$ in this figure refers to a lossy spatial compression algorithm. In several video compression standards (e.g., H.261, MPEG-1 and MPEG-2), this lossy compression algorithm involves partitioning the frame into 8 × 8 blocks (4 DCT blocks per and then quantizing the individual DCT coefficients, to produce indices

$$q_i [n] = \left\langle \frac{y_i [n]}{\Delta_i} \right\rangle$$

Here, $y_i [n]$ is the $i^{th}$ coefficient from DCT block $n$, while $\Delta_i$ is the quantization step size selected for these coefficients and $\langle \cdot \rangle$ denotes rounding to the
nearest integer. \( Q^{-1} \) refers to the “approximate” inverse of this compression algorithm, in which \( y_i [n] \) is reconstructed approximately as \( \bar{y}_i [n] = q_i [n] \Delta_i \) and the inverse DCT is applied to each block of these reconstructed coefficients.

The idea in Figure 11 is to apply the lossy compression algorithm not to the frames themselves, but to the displaced frame difference (DFD) instead, after application of block-based motion compensation. Specifically, each frame, \( f_k \), is partitioned into so-called “macro-blocks” (MB’s), \( B_j \), and a block matching strategy is used to determine backward motion vectors, \( d_{k,j}^b \), for each MB. Accordingly, we expect the DFD to involve values that are mostly close to 0. Let \( \bar{f}_k \) denote the version of frame \( f_k \) which is available to the decompressor. The purpose of the MC feedback loop is to ensure that the compressor reconstructs a copy of the same quantized representation of \( f_k \) that the decompressor will receive so that the next frame, \( f_k \), is coded with respect to the quantized version of the last frame, \( \bar{f}_{k-1} \), which the decompressor has. Specifically, the spatial compression algorithm is applied to the prediction error image (DFD), defined by

\[
E_k[n] = f_k[n] - \tilde{f}_{k-1}(n - d_{k,j}^b), \quad \forall n \in B_j
\]

(13)

The decompressor reconstructs \( \tilde{f}_k \) by motion compensating its quantized version of the last frame, i.e. \( \tilde{f}_{k-1} \), and adding the decompressed prediction error, \( \tilde{E}_k \), i.e.

\[
\tilde{f}_k[n] = \tilde{E}_k[n] + \tilde{f}_{k-1}(n - d_{k,j}^b), \quad \forall n \in B_j
\]

(14)

and the compressor does the same thing in order to keep track with the decompressor.
One major disadvantage of this predictive feedback strategy is that it is highly susceptible to errors. If a transmission error occurs then the compressor and decompressor lose synchronization and the error tends to grow and propagate ad infinitum into future frames. To avoid this difficulty, it is important to periodically turn off the prediction scheme so that the frame pixels, \( f_k[n] \), are coded directly with the lossy spatial compression scheme. This is known as intra-frame coding, as opposed to the predictive inter-frame coding strategy.

### 5.2 Coding Gain for Motion Compensation

We have already seen how the impact of spatial transforms, such as the DCT, can be understood using the concept of coding gain. We can also develop a coding gain expression to reveal the impact of motion compensated prediction. To do this, it is helpful to consider the lossy compression operator \( Q \) of Figure 11 to be independent quantization of every sample of the DFD image \( \mathcal{E}_k[n] \), as opposed to something more sophisticated such as the DCT. In this case, we can write the expected distortion (MSE) in the DFD image as

\[
D \mathcal{E} (R) = \sigma^2_{\mathcal{E}} e^{-aR}
\]

for some \( g \) and \( a \).

Now an important property of the motion compensated feedback loop is that

\[
D \mathcal{E} = E \left[ (\mathcal{E}_k[n] - \bar{\mathcal{E}}_k[n])^2 \right] = E \left[ (F_k[n] - \bar{F}_k[n])^2 \right] = D_{MC}
\]

where we have used equations (13) and (14) to deduce the second line of the above equation. That is, MSE distortion associated with quantizing the DFD is identical to the MSE distortion in the decompressed video.

We can now compare the distortion at rate \( R \) that can be expected with and without motion compensation, by comparing \( D_{MC} (R) = D \mathcal{E} (R) \) with \( D_{PCM} (R) = \sigma^2_X e^{-aR} \), where \( \sigma^2_X \) is the variance of the original video frame samples \( f_k[n] \). It follows that the coding gain can be expressed as

\[
G_{MC} = \frac{D_{PCM} (R)}{D_{MC} (R)} = \frac{\sigma^2_X}{\sigma^2_{\mathcal{E}}}
\]

The smaller the displaced frame difference energy is, the larger the coding gain.
5.3 The MPEG-1 Video Compression Algorithm

The MPEG-1 standard and its successors are developed within the International Standards Organization (ISO) under the auspices of ISO/IEC JTC 1 SC29 WG11 (MPEG started within Working Group 8 and later obtained its own distinct identity). The primary goal for MPEG was to standardize a compression system capable of delivering high quality video (and associated audio) for applications where low end-to-end delay is not a key consideration.

The MPEG-1 standard builds on the simple motion compensated scheme suggested in Figure 11 in the following ways:

- MPEG introduces bi-directional prediction, in addition to the intra-frame and inter-frame predictive coding modes of H.261.

- In MPEG, motion vectors may be expressed as multiples of half the pixel spacing – i.e. the motion vector grid spacing, \( g = \frac{1}{2} \). Each frame contains a flag indicating whether the motion vectors will be specified to full pixel precision (integers), or half pixel precision.

- The frames in an MPEG sequence are grouped into blocks, known as a Group Of Pictures (GOP). The most common GOP size is 15 frames. The GOP contains at least one I-frame, whose MB’s are all intra-frame coded; this is usually the first frame in the GOP. The remaining frames are identified as either P- or B-frames, where the MB’s in each P-frame may be either intra- or inter-frame coded using the motion compensated prediction strategy of H.261 and the MB’s in each B-frame may additionally be coded using bi-directional motion compensated prediction. H.261 does not organize frames into groups or consider any particular frame differently from another.

- MPEG uses explicit quantization tables to specify the \( \Delta_i \) values for each of the 64 DCT coefficients. Separate explicit quantization tables are used for intra-frame and inter-frame coding modes. These two tables are to be used for the entire sequence, except that a global scaling factor is applied to the tables, which may be modified periodically for rate control. This scaling factor is known as the quantization factor. It is explicitly set at the beginning of each “slice”, which is MPEG’s terminology for a group of blocks. The length of a slice is variable; it can range from a single MB to an entire frame. As in H.261, signalling methods are provided to allow the quantization factor to be changed at any MB within a slice, although the use of this facility is more costly.
In the remainder of this section, we elaborate upon the bi-directional motion compensated prediction, which is arguably the most significant departure in MPEG from H.261. Figure 12 illustrates a typical GOP structure with 9 frames in each GOP in this case (15 is more common). As mentioned above, the P-frames are coded in a manner very similar to H.261 so that each MB is either intra-frame coded or inter-frame coded and inter MB’s are coded with reference to the previous I-frame or P-frame, usually after motion compensation. This is known as forward motion compensated prediction, while the relevant motion model uses backward motion estimation with motion vectors, $d_{k,j}^b$, for block $B_j$. In the B-frames, each MB may additionally be coded using bi-directional or a backward motion compensated prediction. Consider the MB, $B_j$, in a B-frame, $f_k$. Let $d_{k,j}^b$ denote the corresponding backward motion vector and $d_{k,j}^f$ denote the forward motion vector. If the MB is coded using forward motion compensated prediction then the forward prediction error which is coded is

$$
\mathcal{E}_k^f[n] = f_k[n] - \bar{f}_{k-p_k}^f(n - d_{k,j}^b)
$$

where $p_k^b > 0$ is the temporal displacement between frame $k$ and the previous P- or I-frame. If the MB is coded using backward motion compensated prediction then the backward prediction error which is coded is

$$
\mathcal{E}_k^b[n] = f_k[n] - \bar{f}_{k+p_k}^b(n + d_{k,j}^f)
$$

where $p_k^f > 0$ is the temporal displacement from frame $k$ to the next P- or I-frame. Finally, if the MB is bi-directionally predicted then the relevant prediction error is

$$
\mathcal{E}_k[n] = \frac{1}{2}(\mathcal{E}_k^b[n] + \mathcal{E}_k^f[n])
$$
Bi-directional prediction overcomes a significant weakness of forward motion compensated prediction in that the object whose surface is represented by the MB might not be visible in the previous frame, either as a result of occlusion or a scene change. Even when the relevant surface is visible in the previous frame, bi-directional prediction generally produces lower prediction residuals because it averages the effects of sampling noise and imperfections in the motion model over two frames. B-frames may not themselves be used as a source of prediction so if the percentage of B-frames in the video sequence becomes too high, the quality of the prediction tends to fall because it must be based on widely separated frames. Typically, 2 or 3 B-frames are interspersed between each pair of non-B-frames.

The I-B-P GOP structure in MPEG provides a mechanism for skipping over compressed frames to facilitate random access and fast forwards capabilities. Random access is available at the GOP level since GOP’s are almost entirely independent: every frame in a GOP may be decoded with reference to at most the last frame of the previous GOP or the first frame of the next GOP, both of which may be reconstructed without reference to any additional GOP’s. Fast forward behaviour may be obtained by skipping the B-frames or, in the extreme, skipping over all but the I-frames.

The main drawback of bi-directional prediction is that frames cannot be encoded in the order in which they arrive; the required reorganization of frames implies an inherent end-to-end delay. Similarly, decoded frames must be reorganized before display, which imposes further delays and memory demands.