Image Data Compression

11.1 INTRODUCTION

Image data compression is concerned with minimizing the number of bits required to represent an image. Perhaps the simplest and most dramatic form of data compression is the sampling of bandlimited images, where an infinite number of pixels per unit area is reduced to one sample without any loss of information (assuming an ideal low-pass filter is available). Consequently, the number of samples per unit area is infinitely reduced.

Applications of data compression are primarily in transmission and storage of information. Image transmission applications are in broadcast television, remote sensing via satellite, military communications via aircraft, radar and sonar, teleconferencing, computer communications, facsimile transmission, and the like. Image storage is required for educational and business documents, medical images that arise in computer tomography (CT), magnetic resonance imaging (MRI) and digital radiology, motion pictures, satellite images, weather maps, geological surveys, and so on. Application of data compression is also possible in the development of fast algorithms where the number of operations required to implement an algorithm is reduced by working with the compressed data.

Image Raw Data Rates

Typical television images have spatial resolution of approximately $512 \times 512$ pixels per frame. At 8 bits per pixel per color channel and 30 frames per second, this translates into a rate of nearly $180 \times 10^6$ bits/s. Depending on the application, digital image raw data rates can vary from $10^5$ bits per frame to $10^8$ bits per frame or higher. The large-channel capacity and memory requirements (see Table 1.1b) for digital image transmission and storage make it desirable to consider data compression techniques.
Data Compression versus Bandwidth Compression

The mere process of converting an analog video signal into a digital signal results in increased bandwidth requirements for transmission. For example, a 4-MHz television signal sampled at Nyquist rate with 8 bits per sample would require a bandwidth of 32 MHz when transmitted using a digital modulation scheme, such as phase shift keying (PSK), which requires 1 Hz per 2 bits. Thus, although digitized information has advantages over its analog form in terms of processing flexibility, random access in storage, higher signal to noise ratio for transmission with the possibility of errorless communication, and so on, one has to pay the price in terms of this eightfold increase in bandwidth. Data compression techniques seek to minimize this cost and sometimes try to reduce the bandwidth of the digital signal below its analog bandwidth requirements.

Image data compression methods fall into two common categories. In the first category, called predictive coding, are methods that exploit redundancy in the data. Redundancy is a characteristic related to such factors as predictability, randomness, and smoothness in the data. For example, an image of constant gray levels is fully predictable once the gray level of the first pixel is known. On the other hand, a white noise random field is totally unpredictable and every pixel has to be stored to reproduce the image. Techniques such as delta modulation and differential pulse code modulation fall into this category. In the second category, called transform coding, compression is achieved by transforming the given image into another array such that a large amount of information is packed into a small number of samples. Other image data compression algorithms exist that are generalizations or combinations of these two methods. The compression process inevitably results in some distortion because of accompanying A to D conversion as well as rejection of some relatively insignificant information. Efficient compression techniques tend to minimize this distortion. For digitized data, distortionless compression techniques are possible. Figure 11.1 gives a summary classification of various data compression techniques.

Information Rates

Raw image data rate does not necessarily represent its average information rate, which for a source with $L$ possible independent symbols with probabilities $p_i$. 

![Figure 11.1 Image data compression techniques.](image-url)
\[ i = 0, \ldots, L - 1, \text{ is given by the entropy} \]

\[
H = - \sum_{i=0}^{L-1} p_i \log_2 p_i \quad \text{bits per symbol} \tag{11.1}
\]

According to Shannon’s noiseless coding theorem (Section 2.13) it is possible to code, without distortion, a source of entropy \( H \) bits per symbol using \( H + \epsilon \) bits per symbol, where \( \epsilon \) is an arbitrarily small positive quantity. Then the maximum achievable compression \( C \), defined by

\[
C = \frac{\text{average bit rate of the original raw data (} B \text{)}}{\text{average bit rate of the encoded data (} H + \epsilon \text{)}} \tag{11.2}
\]

is \( B/(H + \epsilon) = B/H \). Computation of such a compression ratio for images is impractical, if not impossible. For example an \( N \times M \) digital image with \( B \) bits per pixel is one of \( L = 2^{BNM} \) possible image patterns that could occur. Thus if \( p_i \), the probability of the \( i \)th image pattern, were known, one could compute the entropy, that is, the information rate for \( B \) bits per pixel \( N \times M \) images. Then one could store all the \( L \) possible image patterns and encode the image by its address—using a suitable encoding method, which will require approximately \( H \) bits per image or \( H/NM \) bits per pixel.

Such a method of coding is called vector quantization, or block coding [12]. The main difficulty with this method is that even for small values of \( N \) and \( M \), \( L \) can be prohibitively large; for example, for \( B = 8 \), \( N = M = 16 \) and \( L = 2^{2048} \approx 10^{614} \). Figure 11.2 shows a practical adaptation of this idea for vector quantization of \( 4 \times 4 \) image blocks with \( B = 6 \). Each block is normalized to have zero mean and unity variance. Using a few prototype training images, the most probable subset containing \( L' \leq L \) images is stored. If the input block is one of these \( L' \) blocks, it is coded by the address of the block; otherwise it is replaced by its mean value.

The entropy of an image can also be estimated from its conditional entropy. For a block of \( N \) pixels \( u_0, u_1, \ldots, u_{N-1} \), with \( B \) bits per pixel and arranged in an arbitrary order, the \( N \)th-order conditional entropy is defined as

\[
H_N \Delta = -\sum_{u_0} \ldots \sum_{u_{N-1}} p(u_0, u_1, \ldots, u_{N-1}) \log_2 p(u_0 | u_1, \ldots, u_{N-1}) \tag{11.3}
\]

![Figure 11.2 Vector quantization of images.](image-data-compression-chap-11.png)
where each $u_i, i = 0, \ldots, N - 1$, takes $2^8$ values, and $p(\cdot; \cdot, \cdot, \ldots)$ represent the relevant probabilities. For 8-bit monochrome television-quality images, the zero- to second-order entropies (with nearest-neighbor ordering) generally lie in the range of 2 to 6 bits/pixel. Theoretically, for ergodic sequences, as $N \to \infty$, $H_N$ converges to $H$, the per-pixel entropy. Shannon's theory tells us that the bit rate of any exact coding method can never be below the entropy $H$.

**Subsampling, Coarse Quantization, Frame Repetition, and Interlacing**

One obvious method of data compression would be to reduce the sampling rate, the number of quantization levels, and the refresh rate (number of frames per second) down to the limits of aliasing, contouring, and flickering phenomena, respectively. The distortions introduced by subsampling and coarse quantization for a given level of compression are generally much larger than the more sophisticated methods available for data compression. To avoid flicker in motion images, successive frames have to be refreshed above the critical fusion frequency (CFF), which is 50 to 60 pictures per second (Section 3.12). Typically, to capture motion a refresh rate of 25 to 30 frames per second is generally sufficient. Thus, a compression of 2 to 1 could be achieved by transmitting (or storing) only 30 frames per second but refreshing at 60 frames per second by repeating each frame. This requires a frame storage, but an image breakup or jump effect (not flicker) is often observed. Note that the frame repetition rate is chosen at 60 per second rather than 55 per second, for instance, to avoid any interference with the line frequency of 60 Hz (in the United States).

Instead of frame skipping and repetition, line interlacing is found to give better visual rendition. Each frame is divided into an odd field containing the odd line addresses and an even field containing the even line addresses; frames are transmitted alternately. Each field is displayed at half the refresh rate in frames per second. Although the jump or image breakup effect is significantly reduced by line interlacing, spatial frequency resolution is somewhat degraded because each field is a subsampled image. An appropriate increase in the scan rate (that is, lines per frame) with line interlacing gives an actual compression of about 37% for the same subjective quality at the 60 frames per second refresh rate without repetition. The success of this method rests on the fact that the human visual system has poor response for simultaneously occurring high spatial and temporal frequencies. Other interlacing techniques, such as vertical line interlacing in each field (Fig. 4.9), can reduce the data rate further without introducing aliasing if the spatial frequency spectrum does not contain simultaneously horizontal and vertical high frequencies (such as diagonal edges). Interlacing techniques are unsuitable for the display of high resolution graphics and other computer generated images that contain sharp edges and transitions. Such images are commonly displayed on a large raster (e.g., $1024 \times 1024$) refreshed at 60 Hz.

**11.2 PIXEL CODING**

In these techniques each pixel is processed independently, ignoring the inter pixel dependencies.
In PCM the incoming video signal is sampled, quantized, and coded by a suitable code word (before feeding it to a digital modulator for transmission) (Fig. 11.3). The quantizer output is generally coded by a fixed-length binary code word having $B$ bits. Commonly, 8 bits are sufficient for monochrome broadcast or videoconferencing quality images, whereas medical images or color video signals may require 10 to 12 bits per pixel.

The number of quantizing bits needed for visual display of images can be reduced to 4 to 8 bits per pixel by using companding, contrast quantization, or dithering techniques discussed in Chapter 4. Halftone techniques reduce the quantizer output to 1 bit per pixel, but usually the input sampling rate must be increased by a factor of 2 to 16. The compression achieved by these techniques is generally less than $2:1$.

In terms of a mean square distortion, the minimum achievable rate by PCM is given by the rate-distortion formula

$$R_{PCM} = \frac{1}{2} \log_2 \frac{\sigma_u^2}{\sigma_q^2}, \quad \sigma_q^2 < \sigma_u^2$$

where $\sigma_u^2$ is the variance of the quantizer input and $\sigma_q^2$ is the quantizer mean square distortion.

**Entropy Coding**

If the quantized pixels are not uniformly distributed, then their entropy will be less than $B$, and there exists a code that uses less than $B$ bits per pixel. In entropy coding the goal is to encode a block of $M$ pixels containing $MB$ bits with probabilities $p_i, i = 0, 1, \ldots, L - 1, L = 2^{MB}$, by $-\log_2 p_i$ bits, so that the average bit rate is

$$\sum_i p_i (-\log_2 p_i) = H$$

This gives a variable-length code for each block, where highly probable blocks (or symbols) are represented by small-length codes, and vice versa. If $-\log_2 p_i$ is not an integer, the achieved rate exceeds $H$ but approaches it asymptotically with increasing block size. For a given block size, a technique called Huffman coding is the most efficient fixed to variable length encoding method.

**The Huffman Coding Algorithm**

1. Arrange the symbol probabilities $p_i$ in a decreasing order and consider them as leaf nodes of a tree.

![Figure 11.3 Pulse code modulation (PCM).](image-url)
2. While there is more than one node:
   - Merge the two nodes with smallest probability to form a new node whose probability is the sum of the two merged nodes.
   - Arbitrarily assign 1 and 0 to each pair of branches merging into a node.
3. Read sequentially from the root node to the leaf node where the symbol is located.

The preceding algorithm gives the *Huffman code book* for any given set of probabilities. Coding and decoding is done simply by looking up values in a table. Since the code words have variable length, a buffer is needed if, as is usually the case, information is to be transmitted over a constant-rate channel. The size of the code book is $L$ and the longest code word can have as many as $L$ bits. These parameters become prohibitively large as $L$ increases. A practical version of Huffman code is called the *truncated Huffman code*. Here, for a suitably selected $L_1 < L$, the first $L_1$ symbols are Huffman coded and the remaining symbols are coded by a prefix code followed by a suitable fixed-length code.

Another alternative is called the *modified Huffman code*, where the integer $i$ is represented as

$$i = qL_1 + j, \quad 0 \leq q \leq \text{Int}\left[\frac{(L - 1)}{L_1}\right], 0 \leq j \leq L_1 - 1$$

The first $L_1$ symbols are Huffman coded. The remaining symbols are coded by a prefix code representing the quotient $q$, followed by a *terminator code*, which is the same as the Huffman code for the remainder $j$, $0 \leq j \leq L_1 - 1$.

The long-term histogram for television images is approximately uniform, although the short-term statistics are highly nonstationary. Consequently entropy coding is not very practical for raw image data. However, it is quite useful in predictive and transform coding algorithms and also for coding of binary data such as graphics and facsimile images.

**Example 11.1**

Figure 11.4 shows an example of the tree structure and the Huffman codes. The algorithm gives code words that can be uniquely decoded. This is because no code word can be a prefix of any larger-length code word. For example, if the Huffman coded bit stream is

$$0\ 0\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ 1\ \ldots$$

then the symbol sequence is $s_0s_2s_3s_3\ldots$. A prefix code (circled elements) is obtained by reading the code of the leaves that lead up to the first node that serves as a root for the truncated symbols. In this example there are two prefix codes (Fig. 11.4). For the truncated Huffman code the symbols $s_4, \ldots, s_7$ are coded by a 2-bit binary code word. This code happens to be less efficient than the simple fixed length binary code in this example. But this is not typical of the truncated Huffman code.

**Run-Length Coding**

Consider a binary source whose output is coded as the number of 0s between two successive 1s, that is, the length of the runs of 0s are coded. This is called *run-length*
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Binary code</th>
<th>( p_i )</th>
<th>Huffman code (HC)</th>
<th>Truncated Huffman code, ( L_1 = 2 ) (THC)</th>
<th>Modified Huffman code, ( L_1 = 4 ) (MHC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s_0 )</td>
<td>0 0 0</td>
<td>0.25</td>
<td>0 0 0</td>
<td>0 0</td>
<td></td>
</tr>
<tr>
<td>( s_1 )</td>
<td>0 0 1</td>
<td>0.21</td>
<td>0 1 0</td>
<td>1 0</td>
<td></td>
</tr>
<tr>
<td>( s_2 )</td>
<td>0 1 0</td>
<td>0.15</td>
<td>0 29</td>
<td>0 1 0</td>
<td></td>
</tr>
<tr>
<td>( s_3 )</td>
<td>0 1 1</td>
<td>0.14</td>
<td>1</td>
<td>0 1 1</td>
<td></td>
</tr>
<tr>
<td>( s_4 )</td>
<td>1 0 0</td>
<td>0.0625</td>
<td>0.125</td>
<td>1 1 1 0</td>
<td></td>
</tr>
<tr>
<td>( s_5 )</td>
<td>1 0 1</td>
<td>0.0625</td>
<td>0.25</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>( s_6 )</td>
<td>1 1 0</td>
<td>0.0625</td>
<td>0.125</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>( s_7 )</td>
<td>1 1 1</td>
<td>0.0625</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Average code length: 3.0, 2.781 (entropy)
Code efficiency \( H/B_e \): 92.7%, 99.7%

Figure 11.4 Huffman coding example. \( x, x', x'' \) = prefix codes, \( y \) = fixed length code, \( z \) = terminator code. In general, \( x, x', x'' \) can be different.

Coding (RLC). It is useful whenever large runs of 0s are expected. Such a situation occurs in printed documents, graphics, weather maps, and so on, where \( p \), the probability of a 0 (representing a white pixel) is close to unity. (See Section 11.9.)

Suppose the runs are coded in maximum lengths of \( M \) and, for simplicity, let \( M = 2^m - 1 \). Then it will take \( m \) bits to code each run by a fixed-length code. If the successive 0s occur independently, then the probability distribution of the run lengths turns out to be the geometric distribution

\[
g(l) = \begin{cases} 
p^l(1 - p), & 0 \leq l \leq M - 1 \\
p^M, & l = M \end{cases}
\]

Since a run length of \( l \leq M - 1 \) implies a sequence of \( l \) 0s followed by a 1, that is,
(l + 1) symbols, the average number of symbols per run will be
\[ \mu = \sum_{i=0}^{M-1} (l + 1)p^i (1 - p) + Mp^M \]
\[ = \frac{(1 - p^M)}{(1 - p)} \]  \hspace{1cm} (11.7)

Thus it takes \( m \) bits to establish a run-length code for a sequence of \( \mu \) binary symbols, on the average. The compression achieved is, therefore,
\[ C = \frac{\mu}{m} = \frac{(1 - p^M)}{m(1 - p)} \]  \hspace{1cm} (11.8)

For \( p = 0.9 \) and \( M = 15 \) we obtain \( m = 4, \mu = 7.94, \) and \( C = 1.985. \) The achieved average rate is \( B_a = m/\mu = 0.516 \) bit per pixel and the code efficiency, defined as \( H/B_a, \) is 0.469/0.516 = 91\%. For a given value of \( p, \) the optimum value of \( M \) can be determined to give the highest efficiency. RLC efficiency can be improved further by going to a variable length coding method such as Huffman coding for the blocks of length \( m. \) Another alternative is to use arithmetic coding [10] instead of the RLC.

**Bit-Plane Encoding [11]**

A 256 gray-level image can be considered as a set of eight 1-bit planes, each of which can be run-length encoded. For 8-bit monochrome images, compression ratios of 1.5 to 2 can be achieved. This method becomes very sensitive to channel errors unless the significant bit planes are carefully protected.

### 11.3 PREDICTIVE TECHNIQUES

**Basic Principle**

The philosophy underlying predictive techniques is to remove mutual redundancy between successive pixels and encode only the new information. Consider a sampled sequence \( u(m), \) which has been coded up to \( m = n - 1 \) and let \( u^*(n - 1), u^*(n - 2), \ldots \) be the values of the reproduced (decoded) sequence. At \( m = n, \) when \( u(n) \) arrives, a quantity \( \bar{u}(n), \) an estimate of \( u(n), \) is predicted from the previously decoded samples \( u^*(n - 1), u^*(n - 2), \ldots, \) that is,
\[ \bar{u}(n) = \psi(u^*(n - 1), u^*(n - 2), \ldots) \]  \hspace{1cm} (11.9)

where \( \psi(\cdot, \cdot, \ldots) \) denotes the prediction rule. Now it is sufficient to code the prediction error
\[ e(n) = u(n) - \bar{u}(n) \]  \hspace{1cm} (11.10)

If \( e^*(n) \) is the quantized value of \( e(n), \) then the reproduced value of \( u(n) \) is taken as
\[ u^*(n) = \bar{u}(n) + e^*(n) \]  \hspace{1cm} (11.11)

The coding process continues recursively in this manner. This method is called differential pulse code modulation (DPCM) or differential PCM. Figure 11.5 shows
the DPCM codec (coder-decoder). Note that the coder has to calculate the reproduced sequence \( u'(n) \). The decoder is simply the predictor loop of the coder. Rewriting (11.10) as

\[
 u(n) = \bar{u}(n) + e(n)
\]

and subtracting (11.11) from (11.12), we obtain

\[
 \delta u(n) \triangleq u(n) - u'(n) = e(n) - e'(n) = q(n)
\]

Thus, the pointwise coding error in the input sequence is exactly equal to \( q(n) \), the quantization error in \( e(n) \). With a reasonable predictor the mean square value of the differential signal \( e(n) \) is much smaller than that of \( u(n) \). This means, for the same mean square quantization error, \( e(n) \) requires fewer quantization bits than \( u(n) \).

**Feedback Versus Feedforward Prediction**

An important aspect of DPCM is (11.9) which says the prediction is based on the output—the quantized samples—rather than the input—the unquantized samples. This results in the predictor being in the feedback loop around the quantizer, so that the quantizer error at a given step is fed back to the quantizer input at the next step. This has a stabilizing effect that prevents dc drift and accumulation of error in the reconstructed signal \( u'(n) \).

On the other hand, if the prediction rule is based on the past inputs (Fig. 11.6a), the signal reconstruction error would depend on all the past and present

---

**Figure 11.5** Differential pulse code modulation (DPCM) CODEC.

**Figure 11.6** Feedforward coding (a) with distortion; (b) distortionless.
quantization errors in the feedforward prediction-error sequence $e(n)$. Generally, the mean square value of this reconstruction error will be greater than that in DPCM, as illustrated by the following example (also see Problem 11.3).

**Example 11.2**

The sequence 100, 102, 120, 120, 120, 118, 116 is to be predictively coded using the previous element prediction rule, $\tilde{u}(n) = u(n - 1)$ for DPCM and $u(n) = u(n - 1)$ for the feedforward predictive coder. Assume a 2-bit quantizer shown in Fig. 11.7 is used, except the first sample is quantized separately by a 7-bit uniform quantizer, giving $u(0) = u(0) = 100$. The following table shows how reconstruction error builds up with a feedforward predictive coder, whereas it tends to stabilize with the feedback system of DPCM.

<table>
<thead>
<tr>
<th>Input</th>
<th>DPCM</th>
<th>Feedforward Predictive Coder</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>$u(n)$</td>
<td>$\tilde{u}(n)$</td>
</tr>
<tr>
<td>0</td>
<td>100</td>
<td>—</td>
</tr>
<tr>
<td>1</td>
<td>102</td>
<td>100</td>
</tr>
<tr>
<td>Edge→</td>
<td>2</td>
<td>120</td>
</tr>
<tr>
<td>3</td>
<td>120</td>
<td>106</td>
</tr>
<tr>
<td>4</td>
<td>120</td>
<td>111</td>
</tr>
<tr>
<td>5</td>
<td>118</td>
<td>116</td>
</tr>
</tbody>
</table>

**Distortionless Predictive Coding**

In digital processing the input sequence $u(n)$ is generally digitized at the source itself by a sufficient number of bits (typically 8 for images). Then, $u(n)$ may be considered as an integer sequence. By requiring the predictor outputs to be integer values, the prediction error sequence will also take integer values and can be entropy coded without distortion. This gives a distortionless predictive codec (Fig. 11.6b), whose minimum achievable rate would be equal to the entropy of the prediction-error sequence $\epsilon(n)$. 

Sec. 11.3 Predictive Techniques
Performance Analysis of DPCM

Denoting the mean square values of quantization error \( q(n) \) and the prediction error \( e(n) \) by \( \sigma^2_q \) and \( \sigma^2_e \), respectively, and noting that (11.13) implies

\[
E[(\delta u(n))^2] = \sigma^2_q
\]

(11.14)

the minimum achievable rate by DPCM is given by the rate-distortion formula [see (2.116)]

\[
R_{DPCM} = \frac{1}{2} \log_2 \left( \frac{\sigma^2_e}{\sigma^2_q} \right) \text{ bits/pixel}
\]

(11.15)

In deducing this relationship, we have used the fact that common zero memory quantizers (for arbitrary distributions) do not achieve a rate lower than the Shannon quantizer for Gaussian distributions (see Section 4.9). For the same distortion \( \sigma^2_q \leq \sigma^2_e \), the reduction in DPCM rate compared to PCM is [see (11.4)]

\[
R_{PCM} - R_{DPCM} = \frac{1}{2} \log_2 \left( \frac{\sigma^2_e}{\sigma^2_q} \right) = \frac{1}{0.6} \log_{10} \left( \frac{\sigma^2_e}{\sigma^2_q} \right) \text{ bits/pixel}
\]

(11.16)

This shows the achieved compression depends on the reduction of the variance ratio \( (\sigma^2_e/\sigma^2_q) \), that is, on the ability to predict \( u(n) \) and, therefore, on the intersample redundancy in the sequence. Also, the recursive nature of DPCM requires that the predictor be causal. For minimum prediction-error variance, the optimum predictor is the conditional mean \( E[u(n)|u(m), m \leq n - 1] \). Because of the quantizer, this is a nonlinear function and is difficult to determine even when \( u(n) \) is a stationary Gaussian sequence. The optimum feedforward predictor is linear and shift invariant for such sequences, that is,

\[
\bar{u}(n) = \phi(u(n - 1), \ldots) = \sum_k a(k) u(n - k)
\]

(11.17)

If the feedforward prediction error \( e(n) \) has variance \( \beta^2 \), then

\[
\beta^2 \leq \sigma^2_e
\]

(11.18)

This is true because \( \bar{u}(n) \) is based on the quantization noise containing samples \( \{u'(m), m \leq n\} \) and could never be better than \( u(n) \). As the number of quantization levels is increased to infinity, \( \sigma^2_e \) will approach \( \beta^2 \). Hence, a lower bound on the rate achievable by DPCM is

\[
R_{min} = \frac{1}{2} \log_2 \frac{\beta^2}{\sigma^2_q} < R_{DPCM}
\]

(11.19)

When the quantization error is small, \( R_{DPCM} \) approaches \( R_{min} \). This expression is useful because it is much easier to evaluate \( \beta^2 \) than \( \sigma^2_q \) in (11.15), and it can be used to estimate the achievable compression. The SNR corresponding to \( \sigma^2_q \) is given by

\[
(SNR)_{DPCM} = 10 \log_{10} \frac{\sigma^2_u}{\sigma^2_q} = 10 \log_{10} \frac{\sigma^2_u}{\sigma^2_e f(B)} \leq 10 \log_{10} \frac{\sigma^2_u}{\beta^2 f(B)}
\]

(11.20)

where \( f(B) \) is the quantizer mean square distortion function for a unit variance.
input and $B$ quantization bits. For equal number of bits used, the gain in SNR of DPCM over PCM is

$$\text{(SNR)}_{DPCM} - \text{(SNR)}_{PCM} = 10 \log_{10} \left( \frac{\sigma_u^2}{\sigma_e^2} \right) = 10 \log_{10} \left( \frac{\sigma_u^2}{\beta^2} \right) \text{ dB} \quad (11.21)$$

which is proportional to the log of the variance ratio $(\sigma_u^2/\beta^2)$. Using (11.16) we note that the increase in SNR is approximately $6(R_{PCM} - R_{DPCM})$ dB, that is, $6$ dB per bit of available compression.

From these measures we see that the performance of predictive coders depends on the design of the predictor and the quantizer. For simplicity, the predictor is designed without considering the quantizer effects. This means the prediction rule deemed optimum for $\bar{u}(n)$ is applied to estimate $\bar{u}^*(n)$. For example, if $\bar{u}(n)$ is given by (11.17) then the DPCM predictor is designed as

$$\bar{u}^*(n) = \phi(\bar{u}^*(n-1), \bar{u}^*(n-2), \ldots) = \sum_k a(k)u^*(n-k) \quad (11.22)$$

In two (or higher) dimensions this approach requires finding the optimum causal prediction rules. Under the mean square criterion the minimum variance causal representations can be used directly. Note that the DPCM coder remains nonlinear even with the linear predictor of (11.22). However, the decoder will now be a linear filter. The quantizer is generally designed using the statistical properties of the innovations sequence $\epsilon(n)$, which can be estimated from the predictor design. Figure 11.8 shows a typical prediction error signal histogram. Note the prediction

![Figure 11.8 Predictions = error histogram.](image)
error takes large values near the edges. Often, the prediction error is modeled as a zero mean uncorrelated sequence with a Laplacian probability distribution, that is,

\[ p(\varepsilon) = \frac{1}{\beta^2} \exp\left(-\frac{\sqrt{2}}{\beta} |\varepsilon|\right) \]  

where \( \beta^2 \) is its variance. The quantizer is generally chosen to be either the Lloyd-Max (for a constant bit rate at the output) or the optimum uniform quantizer (followed by an entropy coder to minimize the average rate). Practical predictive codecs differ with respect to realizations and the choices of predictors and quantizers. Some of the common classes of predictive codecs for images are described next.

**Delta Modulation**

Delta modulation (DM) is the simplest of the predictive coders. It uses a one-step delay function as a predictor and a 1-bit quantizer, giving a 1-bit representation of the signal. Thus

\[ \bar{u}(n) = u^*(n-1), \quad e(n) = u(n) - u^*(n-1) \]  

A practical DM system, which does not require sampling of the input signal, is shown in Fig. 11.9a. The predictor integrates the quantizer output, which is a sequence of binary pulses. The receiver is a simple integrator. Figure 11.9b shows typical input-output signals of a delta modulator. Primary limitations of delta modulation are (1) slope overload, (2) granularity noise, and (3) instability to channel errors. Slope overload occurs whenever there is a large jump or discontinuity in the signal, to which the quantizer can respond only in several delta steps. Granularity noise is the steplike nature of the output when the input signal is almost constant. Figure 11.10b shows the blurring effect of slope overload near the edges and the granularity effect in the constant gray-level background.

Both of these errors can be compensated to a certain extent by low-pass filtering the input and output signals. Slope overload can also be reduced by increasing the sampling rate, which will reduce the interpixel differences. However, the higher sampling rate will tend to lower the achievable compression. An alternative for reducing granularity while retaining simplicity is to go to a tristate delta modulator. The advantage is that a large number (65 to 85%) of pixels are found to be in the level, or 0, state, whereas the remaining pixels are in the ±1 states. Huffman coding the three states or run-length coding the 0 states with a 2-bit code for the other states yields rates around 1 bit per pixel for different images [14].

The reconstruction filter, which is a simple integrator, is unstable. Therefore, in the presence of channel errors, the receiver output can accumulate large errors. It can be stabilized by attenuating the predictor output by a positive constant \( \rho < 1 \) (called leak). This will, however, not retain the simple realization of Fig. 11.9a.

For delta modulation of images, the signal is generally presented line by line and no advantage is taken of the two-dimensional correlation in the data. When each scan line of the image is represented by a first-order AR process (after subtracting the mean),
\[ u(n) = \rho u(n-1) + \varepsilon(n), \quad E[\varepsilon(n)] = 0, \quad E[\varepsilon(n)\varepsilon(m)] = (1 - \rho^2)\sigma^2 \delta(m-n) \]  

the SNR of the reconstructed signal is given, approximately, by (see Problem 11.4)

\[ (\text{SNR})_{\text{DM}} = 10 \log_{10} \frac{1 - (2\rho - 1)f(1)}{2(1 - \rho)f(1)} \text{ dB} \]  

Assuming the prediction error to be Gaussian and quantized by its Lloyd-Max quantizer and \( \rho = 0.95 \), the SNR is 12.8 dB, which is an 8.4-dB improvement over PCM at 1 bit per pixel. This amounts to a compression of 2.5, or a savings of about 1.5 bits per pixel. Equations (11.25) and (11.26) indicate the SNR of delta modu-
ulation can be improved by increasing $\rho$, which can be done by increasing the sampling rate of the quantizer output. For example, by doubling the sampling rate in this example, $\rho$ will be increased to 0.975, and the SNR will increase by 3 dB. At the same time, however, the data rate is also doubled. Better performance can be obtained by going to adaptive techniques or increasing the number of quantizer bits, which leads to DPCM. In fact, a large number of the ills of delta modulation can be cured by DPCM, thereby making it a more attractive alternative for data compression.

**Line-by-Line DPCM**

In this method each scan line of the image is coded independently by the DPCM technique. Generally, a suitable AR representation is used for designing the pre-
dictor. Thus if we have a $p$th-order, stationary AR sequence (see Section 6.2)

$$u(n) - \sum_{k=1}^{p} a(k)u(n-k) = \epsilon(n), \quad E[(\epsilon(n))^2] = \beta^2$$  \hspace{1cm} (11.27)

the DPCM system equations are

**Predictor:** $\bar{u}^\prime(n) = \sum_{k=1}^{p} a(k)u^\prime(n-k)$ \hspace{1cm} (11.28a)

**Quantizer input:** $e(n) = u(n) - \bar{u}^\prime(n)$, quantizer output = $e^\prime(n)$ \hspace{1cm} (11.28b)

**Reconstruction filter:** $u^\prime(n) = \bar{u}^\prime(n) + e^\prime(n)$ \hspace{1cm} (reproduced output)

$$= \sum_{k=1}^{p} a(k)u^\prime(n-k) + e^\prime(n)$$ \hspace{1cm} (11.28c)

For the first-order AR model of (11.25), the SNR of a $B$-bit DPCM system output can be estimated as (Problem 11.6)

$$(\text{SNR})_{\text{DPCM}} = 10 \log_{10} \left( \frac{(1 - \rho^2) f(B)}{(1 - \rho^2) f(B)} \right) \text{ dB}$$  \hspace{1cm} (11.29)

For $\rho = 0.95$ and a Laplacian density-based quantizer, roughly 8-dB to 10-dB SNR improvement over PCM can be expected at rates of 1 to 3 bits per pixel. Alternatively, for small distortion levels ($f(B) = 0$), the rate reduction over PCM is [see (11.16)]

$$R_{\text{PCM}} - R_{\text{DPCM}} = \frac{1}{2} \log_2 \frac{1}{(1 - \rho^2)} \text{ bits/pixel}$$  \hspace{1cm} (11.30)

This means, for example, the SNR of 6-bit PCM can be achieved by 4-bit line-by-line DPCM for $\rho = 0.97$. Figure 11.10c shows a line-by-line DPCM coded image at 3 bits per pixel.

**Two-Dimensional DPCM**

The foregoing ideas can be extended to two dimensions by using the causal MVRs discussed in chapter 6 (Section 6.6), which define a predictor of the form

$$\bar{u}(m,n) = \sum_{(k,l) \in \tilde{W}_1} a(k,l)u(m-k, n-l)$$  \hspace{1cm} (11.31)

where $\tilde{W}_1$ is a causal prediction window. The coefficients $a(k, l)$ are determined by solving (6.66) for $x = 1$, which minimizes the variance of the prediction error in the image. For common images it has been found that increasing size of $\tilde{W}_1$ beyond the four nearest (causal) neighbors (Fig. 11.11) does not give any appreciable reduction in prediction error variance. Thus for row-by-row scanned images, it is sufficient to consider predictors of the form

$$\bar{u}(m,n) = a_1 u(m-1, n) + a_2 u(m, n-1)$$

$$+ a_3 u(m-1, n-1) + a_4 u(m-1, n+1)$$  \hspace{1cm} (11.32)

Sec. 11.3    Predictive Techniques
Figure 11.11 Pixels (A, B, C, D) used in two-dimensional prediction.

Here \( a_1, a_2, a_3, a_4, \) and \( \beta^2 \) are obtained by solving the linear equations

\[
\begin{align*}
    r(1, 0) &= a_1 r(0, 0) + a_2 r(1, -1) + a_3 r(0, 1) + a_4 r(0, 1) \\
    r(0, 1) &= a_1 r(1, -1) + a_2 r(0, 0) + a_3 r(1, 0) + a_4 r(1, -2) \\
    r(1, 1) &= a_1 r(0, 1) + a_2 r(1, 0) + a_3 r(0, 0) + a_4 r(0, 2) \\
    r(1, -1) &= a_1 r(0, 1) + a_2 r(1, -2) + a_3 r(0, 2) + a_4 r(0, 0)
\end{align*}
\]

(11.33)

\[
\beta^2 = E[e^2 (m, n)]
\]

\[
= r(0, 0) - a_1 r(1, 0) - a_2 r(0, 1) - a_3 r(1, 1) - a_4 r(1, -1)
\]

where \( r(k, l) \) is the covariance function of \( u(m, n) \). In the special case of the separable covariance function of (2.84), we obtain

\[
\begin{align*}
    a_1 &= \rho_1, & a_2 &= \rho_2, & a_3 &= -\rho_1 \rho_2, & a_4 &= 0, \\
    \beta^2 &= \sigma^2 (1 - \rho_1^2)(1 - \rho_2^2)
\end{align*}
\]

(11.34)

Recall from Chapter 6 that unlike the one-dimension case, this solution of (11.33) can give rise to an unstable causal model. This means while the prediction error variance will be minimized (ignoring the quantization effects), the reconstruction filter could be unstable causing any channel error to be amplified greatly at the receiver. Therefore, the predictor has to be tested for stability and, if not stable, it has to be modified (at the cost of either increasing the prediction error variance or increasing the predictor order). Fortunately, for common monochrome image data (such as television images), this problem is rarely encountered.

Given the predictor as just described, the equations for a two-dimensional DPCM system become

**Predictor:** \( \bar{u}(m, n) = a_1 u(m - 1, n) + a_2 u(m, n - 1) \)

(11.35a)

\[
+ a_3 u(m - 1, n - 1) + a_4 u(m - 1, n + 1)
\]

**Quantizer input:** \( e(m, n) = u(m, n) - \bar{u}(m, n) \)

(11.35b)

**Reconstruction filter:** \( u(m, n) = \bar{u}(m, n) + e(m, n) \)

(11.35c)

The performance bounds of this method can be evaluated via (11.19) and (11.20). An example of a two-dimensional DPCM coding at 3 bits per pixel is shown in Fig. 11.10d.
Performance Comparisons

Figure 11.12 shows the theoretical SNR versus bit rate of two-dimensional DPCM of images modeled by (11.34) and (11.35) with $a_4 = 0$. Comparison with one-dimensional line-by-line DPCM and PCM is also shown. Note that delta modulation is the same as 1-bit DPCM in these curves. In practice, two-dimensional DPCM does not achieve quite as much as a 20-dB improvement over PCM, as expected for random fields with parameters of (11.34). This is because the two-dimensional separable covariance model is overly optimistic about the variance of the prediction error. Figure 11.13 compares the coding-error images in one- and
two-dimensional DPCM. The subjective quality of an image and its tolerance to channel errors can be improved by two-dimensional predictors. Generally a 3-bit-per-pixel DPCM coder can give very good quality images. With Huffman coding, the output rate of a 3-bit quantizer in two-dimensional DPCM can be reduced to 2 to 2.5 bits per pixel average.

**Remarks**

Strictly speaking, the predictors used in DPCM are for zero mean data (that is, the dc value is zero). Otherwise, for a constant background \( \mu \), the predicted value

\[
\bar{u}(m, n) = (a_1 + a_2 + a_3 + a_4)\mu
\]

would yield a bias of \((1 - a_1 - a_2 - a_3 - a_4)\mu\), which would be zero only if the sum of the predictor coefficients is unity. Theoretically, this will yield an unstable reconstruction filter (e.g., in delta modulation with no leak). This bias can be minimized by (1) choosing the predictors coefficients whose sum is close to but less than unity, (2) designing the quantizer reconstruction level to be zero for inputs near zero, and (3) tracking the mean of the quantizer output and feeding the bias correction to the predictor.

The quantizer should be designed to limit the three types of degradations, *granularity*, *slope overload*, and *edge-busyness*. Coarsely placed inner levels of the quantizer cause granularity in the flat regions of the image. Slope overload occurs at high-contrast edges where the prediction error exceeds the extreme levels of the quantizer, resulting in blurred edges. Edge-busyness is caused at less sharp edges, where the reproduced pixels on adjacent scan lines have different quantization levels. In the region of edges the optimum mean square quantizer based on Laplacian density for the prediction error sequence turns out to be too companded; that is, the inner quantization steps are too small, whereas the outer levels are too coarse, resulting in edge-busyness. A solution for minimizing these effects is to increase the number of quantizer levels and use an entropy coder for its outputs. This increases the dynamic range and the resolution of the quantizer. The average coder rate will now depend on the relative occurrences of the edges. Another alternative is to incorporate visual properties in the quantizer design using the visibility function [18]. In practice, standard quantizers are optimized iteratively to achieve appropriate subjective picture quality.

In hardware implementations of two-dimensional DPCM, the predictor is often simplified to minimize the number of multiplications per step. With reference to Fig. 11.11, some simplified prediction rules are discussed in Table 11.2.

The choice of prediction rule is also influenced by the response of the reconstruction filter to channel errors. See Section 11.8 for details.

For interlaced image frames, the foregoing design principles are applied to each field rather than each frame. This is because successive fields are \(\frac{1}{60}\) s apart and the intrafield correlations are expected to be higher (in the presence of motion) than the pixel correlations in the de-interlaced adjacent lines.

Overall, DPCM is simple and well suited for real-time (video rate) hardware implementation. The major drawbacks are its sensitivity to variations in image
statistics and to channel errors. Adaptive techniques can be used to improve the compression performance of DPCM. (Channel-error effects are discussed in Section 11.8.)

**Adaptive Techniques**

The performance of DPCM can be improved by adapting the quantizer and predictor characteristics to variations in the local statistics of the image data. Adaptive techniques use a range of quantizing characteristics and/or predictors from which a “current optimum” is selected according to local image properties. To eliminate the overhead due to the adaptation procedure, previously coded pixels are used to determine the mode of operation of the adaptive coder. In the absence of transmission errors, this allows the receiver to follow the same sequence of decisions made at the transmitter. Adaptive predictors are generally designed to improve the subjective image quality, especially at the edges. A popular technique is to use several predictors, each of which performs well if the image is highly correlated in a certain direction. The direction of maximum correlation is computed from previously coded pixels and the corresponding predictor is chosen.

Adaptive quantization schemes are based on two approaches, as discussed next.

**Adaptive gain control.** For a fixed predictor, the variance of the prediction error will fluctuate with changes in spatial details of the image. A simple adaptive quantizer updates the variance of the prediction error at each step and adjusts the spacing of the quantizer levels accordingly. This can be done by normalizing the prediction error by its updated standard deviation and designing the quantizer levels for unit variance inputs (Fig. 11.14a).

Let $\sigma_e^2(j)$ and $\sigma^2(j)$ denote the variances of the quantizer input and output, respectively, at step $j$ of a DPCM loop. (For a two-dimensional system, this means we are mapping $(m, n)$ into $j$.) Since $e'(j)$ is available at the transmitter as well as

![Diagram](image)

**Figure 11.14 Adaptive quantization.**
the receiver, it is easy to estimate $\sigma^2(j)$. A simple estimate, called the exponential average variance estimator, is of the form

$$\sigma^2(j + 1) = (1 - \gamma)[e'(j)]^2 + \gamma \sigma^2(j), \quad \sigma^2(0) = (e'(0))^2, \quad j = 0, 1, \ldots \quad (11.37)$$

where $0 \leq \gamma \leq 1$. For small quantization errors, we may use $\sigma^2(j)$ as an estimate of $\sigma^2(j)$. For Lloyd-Max quantizers, since the variance of the input equals the sum of the variances of the output and the quantization error [see (4.47)], we can obtain the recursion for $\sigma^2(j)$ as

$$\sigma^2(j + 1) = \frac{1 - \gamma}{1 - f(B)}[e'(j)]^2 + \gamma \sigma^2(j), \quad j = 0, 1, \ldots \quad (11.38)$$

In practice the estimate of (11.37) may be replaced by

$$\sigma^2(j) = \gamma \sum_{m=1}^{N} |e'(j - m)| \quad (11.39)$$

where $\gamma$ is a constant determined experimentally so that the mean square error is minimized. The above two estimates become poor at low rates, for example, when $B = 1$. An alternative, originally suggested for adaptive delta modulation [7], is to define a gain $G = g(m, n)$, which is recursively updated as

$$g(m, n) = \sum_{(k,l) \in W} \alpha_{k,l} g(m - k, n - l) M(|q_{m-k,n-l}|), \quad \text{with} \quad g_{\text{min}} \leq g(m, n) \leq g_{\text{max}} \quad (11.40)$$

where $M(|q_i|)$ is a multiplier factor that depends on the quantizer levels $q_i$ and $\alpha_{k,l}$ are weights which sum up to unity. Often $\alpha_{k,l} = 1/N_w$, where $N_w$ is the number of pixels in the causal window $W$. For example (see Table 11.1), for a three-level quantizer ($L = 3$) using the predictor neighbors of Fig. 11.11 and the gain-control formula

$$g(m, n) = \frac{1}{2}[g(m - 1, n) M(|q_{m-1,n}|) + g(m, n - 1) M(|q_{m,n-1}|)] \quad (11.41)$$

the multiplier factor $M(|q|)$ takes the values $M(0) = 0.7$, $M(\pm q_1) = 1.7$. The values in Table 11.1 are based on experimental studies [19] on 8-bit images.

**Adaptive classification.** Adaptive classification schemes segment the image into different regions according to spatial detail, or activity, and different quantizer characteristics are used for each activity class (Fig. 11.14b). A simple

**TABLE 11.1 Gain-Control Parameters for Adaptive Quantization in DPCM**

<table>
<thead>
<tr>
<th>$L$</th>
<th>$g_{\text{min}}$</th>
<th>$g_{\text{max}}$</th>
<th>$q = 0$</th>
<th>$\pm q_1$</th>
<th>$\pm q_2$</th>
<th>$\pm q_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>5</td>
<td>55</td>
<td>0.7</td>
<td>1.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>40</td>
<td>0.8</td>
<td>1.0</td>
<td>2.6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>32</td>
<td>0.6</td>
<td>1.0</td>
<td>1.5</td>
<td>4.0</td>
</tr>
</tbody>
</table>
measure of activity is the variance of the pixels in the neighborhood of the pixel to be predicted. The flat regions are quantized more finely than edges or detailed areas. This scheme takes advantage of the fact that noise visibility decreases with increased activity. Typically, up to four activity classes are sufficient. An example would be to divide the image into $16 \times 16$ blocks and classify each block into one of four classes. This requires only a small overhead of 2 bits per block of 256 pixels.

**Other Methods** [17, 20]

At low bit rates ($B = 1$) the performance of DPCM deteriorates rapidly. One reason is that the predictor and the quantizer, which were designed independently, no longer operate at near-optimum levels. Thus the successive inputs to the quan-

<table>
<thead>
<tr>
<th>TABLE 11.2 Summary of Predictive Coding</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Predictor</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear mean square</td>
<td>Predictors of orders 3 to 4 are adequate.</td>
</tr>
<tr>
<td>Previous element $\gamma A$</td>
<td>Determined from image correlations. Performs very well as long as image class does not change very much.</td>
</tr>
<tr>
<td>Averaged prediction</td>
<td>Sharp vertical or diagonal edges are blurred and exhibit edge-busyness. Channel error manifests itself as a horizontal streak.</td>
</tr>
<tr>
<td>a. $\gamma\left(\frac{A + D}{2}\right)$</td>
<td>Significant improvement over previous element prediction for vertical and most sloping edges. Horizontal and gradual rising edges blurred. The two predictors using pixel $D$ perform equally well but better than $(A + C)/2$ on gradual rising edges. Edge-busyness and sensitivity to channel errors much reduced (Fig. 11.38).</td>
</tr>
<tr>
<td>b. $\gamma\left(\frac{A + C}{2}\right)$</td>
<td></td>
</tr>
<tr>
<td>c. $\gamma\left(\frac{A + (C + D)/2}{2}\right)$</td>
<td></td>
</tr>
<tr>
<td>Planar prediction</td>
<td>Better than previous element prediction but worse than averaged prediction with respect to edge busyness and channel errors (Fig. 11.38).</td>
</tr>
<tr>
<td>a. $\gamma(A + (C - B))$</td>
<td></td>
</tr>
<tr>
<td>b. $\gamma\left(A + \frac{D - B}{2}\right)$</td>
<td></td>
</tr>
<tr>
<td>Leak ($\gamma$)</td>
<td>$0 &lt; \gamma &lt; 1$. As the leak is increased, transmission errors become less visible, but granularity and contouring become more visible.</td>
</tr>
<tr>
<td>Quantizer</td>
<td>Recommended when the compression ratio is not too high ($\leq 3$) and a fixed length code is used. Prediction error may be modeled by Laplacian or Gaussian probability densities.</td>
</tr>
<tr>
<td>a. Optimum mean square (Lloyd-Max)</td>
<td></td>
</tr>
<tr>
<td>b. Visual</td>
<td>Difficult to design. One alternative is to perturb the levels of the max quantizer to obtain an increased subjective quality.</td>
</tr>
<tr>
<td>c. Uniform</td>
<td>Useful in high-compression schemes ($&gt;3$) where the quantizer output is entropy coded.</td>
</tr>
</tbody>
</table>
tizer may have significant correlation, and the predictor may not be good enough. Two methods that can improve the performance are

1. Delayed predictive coding
2. Predictive vector quantization

In the first method [17], a tree code is generated by the prediction filter excited by different quantization levels. As successive pixels are coded, the predictor selects a path in the tree (rather than a branch value, as in DPCM) such that the mean square error is minimized. Delays are introduced in the predictor to enable development of a tree with sufficient look-ahead paths.

In the second method [20], the successive inputs to the quantizer are entered in a shift register, whose state is used to define the quantizer output value. Thus the quantizer current output depends on its previous outputs.

11.4 TRANSFORM CODING THEORY

The Optimum Transform Coder

Transform coding, also called block quantization, is an alternative to predictive coding. A block of data is unitarily transformed so that a large fraction of its total energy is packed in relatively few transform coefficients, which are quantized independently. The optimum transform coder is defined as the one that minimizes the mean square distortion of the reproduced data for a given number of total bits. This turns out to be the KL transform.

Suppose an \( N \times 1 \) random vector \( \mathbf{u} \) with zero mean and covariance \( \mathbf{R} \) is linearly transformed by an \( N \times N \) (complex) matrix \( \mathbf{A} \), not necessarily unitary, to produce a (complex) vector \( \mathbf{v} \) such that its components \( v(k) \) are mutually uncorrelated (Fig. 11.15). After quantizing each component \( v(k) \) independently, the output vector \( \mathbf{v}^* \) is linearly transformed by a matrix \( \mathbf{B} \) to yield a vector \( \mathbf{u}^* \). The problem is to find the optimum matrices \( \mathbf{A} \) and \( \mathbf{B} \) and the optimum quantizers such that the overall average mean square distortion

\[
D = \frac{1}{N} E \left[ \sum_{n=1}^{N} (u(n) - u^*(n))^2 \right] = \frac{1}{N} E [ (\mathbf{u} - \mathbf{u}^*)^T (\mathbf{u} - \mathbf{u}^*) ]
\]  

is minimized. The solution of this problem is summarized as follows:

1. For an arbitrary quantizer the optimal reconstruction matrix \( \mathbf{B} \) is given by

\[
\mathbf{B} = \mathbf{A}^{-1} \Gamma
\]

where \( \Gamma \) is a diagonal matrix of elements \( \gamma_k \) defined as

\[
\gamma_k \triangleq \frac{\hat{\lambda}_k}{\lambda_k}
\]

\[
\hat{\lambda}_k \triangleq E [v(k)v^*(k)], \quad \lambda_k \triangleq E [|v(k)|^2]
\]
2. The Lloyd-Max quantizer for each \( v(k) \) minimizes the overall mean square error giving
\[
\Gamma = I \tag{11.45}
\]

3. The optimal decorrelating matrix \( A \) is the KL transform of \( u \), that is, the rows of \( A \) are the orthonormalized eigenvectors of the autocovariance matrix \( R \). This gives
\[
B = A^{-1} = A^{*T} \tag{11.46}
\]

**Proofs**

1. In terms of the transformed vectors \( v \) and \( v' \), the distortion can be written as
\[
D = \frac{1}{N} E \{ Tr [ A^{-1} v - B v'] [ A^{-1} v - B v']^{*T} \} \tag{11.47a}
\]
\[
= \frac{1}{N} Tr [ A^{-1} \Lambda (A^{-1})^{*T} + B \Lambda' B^{*T} - A^{-1} \tilde{\Lambda} B^{*T} - B \tilde{\Lambda}^{*T} (A^{-1})^{*T} ] \tag{11.47b}
\]
where
\[
\Lambda \triangleq E[ vv^{*T} ], \quad \Lambda' \triangleq E[ v'(v')^{*T} ], \quad \tilde{\Lambda} \triangleq E[ v(v')^{*T} ] \tag{11.47c}
\]
Since \( v(k), k = 0, 1, \ldots, N - 1 \) are uncorrelated and are quantized independently, the matrices \( \Lambda, \Lambda', \) and \( \tilde{\Lambda} \) are diagonal with \( \lambda_k, \lambda'_k, \) and \( \tilde{\lambda}_k \) as their respective diagonal elements. Minimizing \( D \) by differentiating it with respect to \( B^* \) (or \( B \)), we obtain (see Problem 2.15)
\[
B \Lambda' - A^{-1} \tilde{\Lambda} = 0 \quad \Rightarrow \quad B = A^{-1} \tilde{\Lambda} (\Lambda')^{-1} \tag{11.48}
\]
which gives (11.43) and
\[ D = \frac{1}{N} \text{Tr}[A^{-1} E[v - \Gamma v'][(v - \Gamma v')^*] (A^{-1})^*] \]  
(11.49a)

2. The preceding expression for distortion can also be written in the form
\[ D = \frac{1}{N} \sum_{i=0}^{N-1} \sum_{k=0}^{N-1} |[A^{-1}]_{i,k}|^2 \hat{\sigma}_q^2(k) \lambda_k \]  
(11.49b)
where \( \hat{\sigma}_q^2(k) \) is the distortion as if \( v(k) \) had unity variance, that is
\[ \hat{\sigma}_q^2(k) = \frac{E[|v(k) - \gamma_k v'(k)|^2]}{\lambda_k} \]  
(11.49c)
From this it follows that \( \hat{\sigma}_q^2(k) \) should be minimized for each \( k \) by the quantizer no matter what \( A \) is.† This means we have to minimize the mean square error between the quantizer input \( v(k) \) and its scaled output \( \gamma_k v'(k) \). Without loss of generality, we can absorb \( \gamma_k \) inside the quantizer and require it to be a minimum mean square quantizer. For a given number of bits, this would be the Lloyd-Max quantizer. Note that \( \gamma_k \) becomes unity for any quantizer whose output levels minimize the mean square quantization error regardless of its decision levels. For the Lloyd-Max quantizer, it is not only that \( \gamma_k \) equals unity, but also its decision levels are such that the mean square quantization error is minimum. Thus (11.45) is true and we get \( B = A^{-1} \). This gives
\[ \sigma_q^2(k) = \lambda_k \hat{\sigma}_q^2(k) = E[|v(k) - v'(k)|^2] = \lambda_k f(n_k) \]  
(11.50)
where \( f(x) \) is the distortion-rate function of an \( x \)-bit Lloyd-Max quantizer for unity variance inputs (Table 4.4). Substituting (11.50) in (11.49b), we obtain
\[ D = \frac{1}{N} \text{Tr}[A^{-1} FA(A^{-1})^*], \quad F \triangleq \text{Diag}\{f(n_k)\} \]  
(11.51)
Since \( v \) equals \( Au \), its covariance is given by the diagonal matrix
\[ E[vv^*] \triangleq \Lambda = ARA^* \]  
(11.52)
Substitution for \( \Lambda \) in (11.51) gives
\[ D = \frac{1}{N} \text{Tr}[A^{-1} FA] \]  
(11.53)
where \( F \) and \( R \) do not depend on \( A \). Minimizing \( D \) with respect to \( A \), we obtain (see Problem 2.15)
\[ 0 = \frac{\partial D}{\partial A} = -\frac{1}{N} [A^{-1} F A R^{-1}]^* + \frac{1}{N} [R^{-1} F]^* \]  
(11.54)
\[ \Rightarrow F(A R^{-1}) = (A R^{-1})F \]  
Thus, \( F \) and \( A R^{-1} \) commute. Because \( F \) is diagonal, \( A R^{-1} \) must also be diagonal. But \( A R^{-1} F \) is also diagonal. Therefore, these two matrices must be related by a diagonal matrix \( G \), as

†Note that \( \hat{\sigma}_q^2(k) \) is independent of the transform \( A \).
ARA* T = (ARA -1)G (11.55)

This implies AA* T = G, so the columns of A must be orthogonal. If A is replaced by G 1/2 A, the overall result of transform coding will remain unchanged because B = A -1. Therefore, A can be taken as a unitary matrix, which proves (11.46). This result and (11.52) imply that A is the KL transform of u (see Sections 2.9 and 5.11).

Remarks

Not being a fast transform in general, the KL transform can be replaced either by a fast unitary transform, such as the cosine, sine, DFT, Hadamard, or Slant, which is not a perfect decorrelator, or by a fast decorrelating transform, which is not unitary. In practice, the former choice gives better performance (Problem 11.9).

The foregoing result establishes the optimality of the KL transform among all decorrelating transformations. It can be shown that it is also optimal among all the unitary transforms (see Problem 11.8) and also performs better than DPCM (which can be viewed as a nonlinar transform; Problem 11.10).

Bit Allocation and Rate-Distortion Characteristics

The transform coefficient variances are generally unequal, and therefore each requires a different number of quantizing bits. To complete the transform coder design we have to allocate a given number of total bits among all the transform coefficients so that the overall distortion is minimum. Referring to Fig. 11.15, for any unitary transform A, arbitrary quantizers, and B = A -1 = A* T; the distortion becomes

\[ D = \frac{1}{N} \sum_{k=0}^{N-1} E[(v(k) - \nu(k))^2] = \frac{1}{N} \sum_{k=0}^{N-1} \sigma_k^2 f(n_k) \]  (11.56)

where \( \sigma_k^2 \) is the variance of the transform coefficient \( v(k) \), which is allocated \( n_k \) bits, and \( f(\cdot) \), the quantizer distortion function, is monotone convex with \( f(0) = 1 \) and \( f(\infty) = 0 \). We are given a desired average bit rate per sample, \( B \); then the rate for the A-transform coder is

\[ R_A \triangleq \frac{1}{N} \sum_{k=0}^{N-1} n_k = B \]  (11.57)

The bit allocation problem is to find \( n_k \geq 0 \) that minimize the distortion \( D \), subject to (11.57). Its solution is given by the following algorithm.

Bit Allocation Algorithm

Step 1. Define the inverse function of \( f'(x) \triangleq df(x)/dx \) as \( h(x) \triangleq f'^{-1}(x) \), or \( h(f'(x)) = x \). Find \( \theta \), the root of the nonlinear equation

\[ R_A \triangleq \frac{1}{N} \sum_{k: \sigma_k^2 > \theta} h \left( \frac{\theta f'(0)}{\sigma_k^2} \right) = B \]  (11.58)
The solution may be obtained by an iterative technique such as the Newton method. The parameter $\theta$ is a threshold that controls which transform coefficients are to be coded for transmission.

**Step 2.** The number of bits allocated to the $k$th transform coefficient are given by

$$n_k = \begin{cases} 0, & \sigma_k^2 < \theta \\ h(\theta f'(0)/\sigma_k^2), & \sigma_k^2 \geq \theta \end{cases}$$

(11.59)

Note that the coefficients whose mean square value falls below $\theta$ are not coded at all.

**Step 3.** The minimum achievable distortion is then

$$D = \frac{1}{N} \left[ \sum_{k : \sigma_k^2 \geq \theta} \sigma_k^2 f(n_k) + \sum_{k : \sigma_k^2 < \theta} \sigma_k^2 \right]$$

(11.60)

Sometimes we specify the average distortion $D = d$ rather than the average rate $R$. In that case (11.60) is first solved for $\theta$. Then (11.59) and (11.58) give the bit allocation and the minimum achievable rate. Given $n_k$, the number of quantizer levels can be approximated as $\text{Int}[2^n]$. Note that $n_k$ is not necessarily an integer.

This algorithm is also useful for calculating the rate versus distortion characteristics of a transform coder based on a given transform $A$ and a quantizer with distortion function $f(x)$.

In the special case of the Shannon quantizer, we have $f(x) = 2^{-2x}$, which gives

$$f'(x) = -(2 \log_2 2) 2^{-2x} \implies h(x) = -\frac{1}{2} \log_2 \left( \frac{-x}{2 \log_2 2} \right)$$

$$n_k = \max \left\{ 0, \frac{1}{2} \log_2 \left( \frac{\sigma_k^2}{\theta} \right) \right\}$$

(11.61)

$$D = \frac{1}{N} \left[ \sum_{\sigma_k^2 \geq \theta} \sigma_k^2 + \sum_{\sigma_k^2 < \theta} \theta \right] = \frac{1}{N} \sum_{k = 0}^{N-1} \min(\theta, \sigma_k^2)$$

(11.62)

$$R_A = \frac{1}{N} \left[ \sum_{\sigma_k^2 \leq \theta} 0 + \sum_{\sigma_k^2 > \theta} \frac{1}{2} \log_2 \left( \frac{\sigma_k^2}{\theta} \right) \right]$$

(11.63)

$$= \frac{1}{N} \sum_{k = 0}^{N-1} \max \left\{ 0, \frac{1}{2} \log_2 \left( \frac{\sigma_k^2}{\theta} \right) \right\}$$

More generally, when $f(x)$ is modeled by piecewise exponentials as in Table 4.4, we can similarly obtain the bit allocation formulas [32]. Equations (11.62) and (11.63) give the rate-distortion bound for transform coding of an $N \times 1$ Gaussian random vector $\mathbf{u}$ by a unitary transform $A$. This means for a fixed distortion $D$, the rate $R_A$ will be lower than the rate achieved by using any practical quantizer. When $D$ is small enough so that $0 < \theta < \min_k \{ \sigma_k^2 \}$, we get $\theta = D$, and

$$R_A = \frac{1}{N} \sum_{k = 0}^{N-1} \frac{1}{2} \log_2 \frac{\sigma_k^2}{D} = \frac{1}{2N} \log_2 \left( \prod_{k = 0}^{N-1} \sigma_k^2 \right) - \frac{1}{2} \log_2 D$$

(11.64)
In the case of the KL transform, \( \sigma_k^2 = \lambda_k \) and \( \prod_k \lambda_k = |R| \), which gives

\[
R_{KL} = \frac{1}{2N} \log_2 |R| - \frac{1}{2} \log_2 D, \quad D < \min_k \{\lambda_k\} \tag{11.65}
\]

For small but equal distortion levels,

\[
R_A - R_{KL} = \frac{1}{2N} \log_2 \left( \prod_{k=0}^{N-1} \sigma_k^2 \right) / |R| \geq 0 \tag{11.66}
\]

where we have used (2.43) to give

\[
|R| = |ARA^{*T}| = \prod_{k=0}^{N-1} [ARA^{*T}]_{kk} = \prod_{k=0}^{N-1} \sigma_k^2
\]

For PCM coding, it is equivalent to assuming \( A = I \), so that

\[
R_{PCM} - R_{KL} = -\frac{1}{2N} \log_2 |\hat{R}| \geq 0 \tag{11.67}
\]

where \( \hat{R} = \{r(m, n)/\sigma_m^2\} \) is the correlation matrix of \( u \), and \( \sigma_m^2 \) are the variances of its elements.

**Example 11.3**

The determinant of the covariance matrix \( R = \{\rho^{m-n}\} \) of a Markov sequence of length \( N \) is \( |R| = (1 - \rho^2)^{N-1} \). This gives

\[
R_{KL} = \frac{N-1}{2N} \log_2 (1 - \rho^2) - \frac{1}{2} \log_2 D, \quad D < \min_k \{\lambda_k\} \tag{11.68}
\]

For \( N = 16 \) and \( \rho = 0.95 \), the value of \( \min_k \{\lambda_k\} \) is 0.026 (see Table 5.2). So for \( D = 0.01 \), we get \( R_{KL} = 1.81 \) bits per sample. Rearranging (11.68) we can write

\[
R_{KL} = \frac{1}{2} \log_2 \left( \frac{1 - \rho^2}{D} \right) - \frac{1}{2N} \log_2 (1 - \rho^2) \tag{11.69}
\]

As \( N \to \infty \), the rate \( R_{KL} \) goes down to a lower bound \( R_{KL}(\infty) = \frac{1}{2} \log_2 (1 - \rho^2)/D \), and \( R_{PCM} - R_{KL}(\infty) = -\frac{1}{2} \log_2 (1 - \rho^2) = 1.6 \) bits per sample. Also, as \( N \to \infty \), the eigenvalues of \( R \) follow the distribution \( \lambda(\omega) = (1 - \rho^2)/(1 + \rho^2 + 2\rho \cos \omega) \), which gives \( \min_k \{\lambda_k\} = (1 - \rho^2)/(1 + \rho^2) = (1 - \rho)/(1 + \rho) \). For \( \rho = 0.95 \), \( D = 0.01 \) we obtain \( R_{KL}(\infty) = 1.6 \) bits per sample.

**Integer Bit Allocation Algorithm.** The number of quantizing bits \( n_k \) are often specified as integers. Then the solution of the bit allocation problem is obtained by applying a theory of marginal analysis [6, 21], which yields the following simple algorithm.

**Step 1.** Start with the allocation \( n_k^0 = 0, 0 \leq k \leq N - 1 \). Set \( j = 1 \).

**Step 2.** \( n_k = n_k^{j-1} + \delta(k - i) \), where \( i \) is any index for which

\[
\Delta_k \overset{\Delta}{=} \sigma_k^2 \left[ f(n_k^{j-1}) - f(n_k^{j-1} + 1) \right]
\]

is maximum. \( \Delta_k \) is the reduction in distortion if the \( j \)th bit is allocated to the \( k \)th quantizer.

Sec. 11.4 Transform Coding Theory 503
Step 3. If $\sum_k n_k \geq NB$, stop; otherwise $j \rightarrow j + 1$ and go to Step 2.

If ties occur for the maximizing index, the procedure is successively initiated with the allocation $n_k = n_k^{-1} + \delta(i - k)$ for each $i$. This algorithm simply means that the marginal returns

$$\Delta_{k,i} \triangleq \sigma_k^2 [f(n_k) - f(n_k^{-1})], \quad k = 0, \ldots, N - 1, j = 1, \ldots, NB \quad (11.70)$$

are arranged in a decreasing order and bits are assigned one by one according to this order. For an average bit rate of $B$, we have to search $N$ marginal returns $NB$ times. This algorithm can be speeded up whenever the distortion function is of the form $f(x) = a2^{-bx}$. Then $\Delta_{k,i} = (1 - 2^{-b})\sigma_k^2 f(n_k^{-1})$, which means the quantizer having the largest distortion, at any step $j$, is allocated the next bit. Thus, as we allocate a bit, we update the quantizer distortion and the step 2 of the algorithm becomes:

Step 2: Find the index $i$ such that

$$D_i = \max_k [\sigma_k^2 f(n_k^{-1})]$$

Then

$$n_k = n_k^{-1} + \delta(k - i)$$

$$D_i = 2^{-b} D_i$$

The piecewise exponential models of Table 4.4 can be used to implement this step.

### 11.5 TRANSFORM CODING OF IMAGES

The foregoing one-dimensional transform coding theory can be easily generalized to two dimensions by simply mapping a given $N \times M$ image $u(m, n)$ to a one-dimensional $NM \times 1$ vector $u$. The KL transform of $u$ would be a matrix of size $NM \times NM$. In practice, this transform is replaced by a separable fast transform such as the cosine, sine, Fourier, Slant, or Hadamard; these, as we saw in chapter 5, pack a considerable amount of the image energy in a small number of coefficients.

To make transform coding practical, a given image is divided into small rectangular blocks, and each block is transform coded independently. For an $N \times M$ image divided into $NM/pq$ blocks, each of size $p \times q$, the main storage requirements for implementing the transform are reduced by a factor of $NM/pq$. The computational load is reduced by a factor of $\log_2 MN / \log_2 pq$ for a fast transform requiring $aN \log_2 N$ operations to transform an $N \times 1$ vector. For $512 \times 512$ images divided into $16 \times 16$ blocks, these factors are 1024 and 2.25, respectively. Although the operation count is not greatly reduced, the complexity of the hardware for implementing small-size transforms is reduced significantly. However, smaller block sizes yield lower compression, as shown by Fig. 11.16. Typically, a block size of $16 \times 16$ is used.

**Two-Dimensional Transform Coding Algorithm.** We now state a practical transform coding algorithm for images (Fig. 11.17).
Figure 11.16 Rate achievable by block KL transform coders for Gaussian random fields with separable covariance function, $p = q = 0.95$, at distortion $D = 0.25\%$.

Figure 11.17 Two-dimensional transform coding.

1. Divide the given image. Divide the image into small rectangular blocks of size $p \times q$ and transform each block to obtain $V_i, i = 0, \ldots, I - 1, I = NM/pq$.

2. Determine the bit allocation. Calculate the transform coefficient variances $\sigma_{k,l}^2$ via (5.36) or Problem 5.29b if the image covariance function is given. Alternatively, estimate the variances $\hat{\sigma}_{k,l}^2$ from the ensemble of coefficients $v_i(k, l)$, $i = 0, \ldots, I - 1$, obtained from a given prototype image normalized to have unity variance. From this, the $\sigma_{k,l}^2$ for the image with variance $\sigma^2$ are estimated.
as \( \sigma^2_{k,l} = \hat{\sigma}^2_{k,l} \sigma^2 \). The \( \hat{\sigma}^2_{k,l} \) can be interpreted as the power spectral density of the image blocks in the chosen transform domain.

The bit allocation algorithms of the previous section can be applied after mapping \( \sigma^2_{k,l} \) into a one-dimensional sequence. The ideal case, where \( f(x) = 2^{-2x} \), yields the formulas

\[
\begin{align*}
n_{k,l} &= \max \left( 0, \frac{1}{2} \log_2 \frac{\sigma^2_{k,l}}{\theta} \right), \\
D &= \frac{1}{pq} \sum_{k=0}^{p-1} \sum_{l=0}^{q-1} \min(\theta, \sigma^2_{k,l}), \\
R_A &= \frac{1}{pq} \sum_{k=0}^{p-1} \sum_{l=0}^{q-1} n_{k,l}(\theta)
\end{align*}
\] (11.71)

Alternatively, the integer bit allocation algorithm can be used. Figure 11.18 shows a typical bit allocation for 16 \times 16 block coding of an image by the cosine transform to achieve an average rate \( B = 1 \) bit per pixel.

3. **Design the quantizers.** For most transforms and common images (which are nonnegative) the dc coefficient \( v_i(0,0) \) is nonnegative, and the remaining coefficients have zero mean values. The dc coefficient distribution is modeled by the Rayleigh density (see Problem 4.15). Alternatively, one-sided Gaussian or Laplacian densities can be used. For the remaining transform coefficients, Laplacian or Gaussian densities are used to design their quantizers. Since the transform coefficients are allocated unequal bits, we need a different quantizer for each value of \( n_{k,l} \). For example, in Fig. 11.18 the allocated bits range from 1 to 8. Therefore, eight different quantizers are needed. To implement these quantizers, the input sample \( v_i(k,l) \) is first normalized so that it has unity variance, that is,

\[
\hat{v}_i(k,l) = \frac{v_i(k,l)}{\sigma_{k,l}}, \quad (k, l) \neq (0,0)
\] (11.72)

These coefficients are quantized by an \( n_{k,l} \)-bit quantizer, which is designed for zero mean, unity variance inputs. Coefficients that are allocated zero bits are
not processed at all. At the decoder, which knows the bit allocation table in advance, the unprocessed coefficients are replaced by zeros (that is, their mean values).

4. **Code the quantizer output.** Code the output into code words and transmit or store.

5. **Reproduce the coefficients.** Assuming a noiseless channel, reproduce the coefficients at the decoder as

\[
v_i^* (k, l) = \begin{cases} 
  v_i (k, l) \sigma_{k, i}, & (k, l) \in I_i \\
  0, & \text{otherwise}
\end{cases} 
\]  

(11.73)

where \( I_i \) denotes the set of transmitted coefficients. The inverse transformation \( U_i^* = A^* T V_i^* A^* \) gives the reproduced image blocks.

Once a bit assignment for transform coefficients has been determined, the performance of the coder can be estimated by the relations

\[
D = \frac{1}{pq} \sum_{k=0}^{p-1} \sum_{l=0}^{q-1} \sigma^2_{k, l} f(n_{k, l}) \quad R_A = \frac{1}{pq} \sum_{k=0}^{p-1} \sum_{l=0}^{q-1} n_{k, l} 
\]  

(11.74)

**Transform Coding Performance Trade-Offs and Examples**

**Example 11.4 Choice of transform**

Figure 11.19 compares the performance of different transforms for 16 \( \times \) 16 block coding of a random field. Table 11.3 shows examples of SNR values at different rates. The cosine transform performance is superior to the other fast transforms and is almost indistinguishable from the KL transform. Recall from Section 5.12 that the cosine transform has near-optimal performance for first-order stationary Markov sequences with \( \rho > 0.5 \). Considerations for the choice of other transforms are summarized in Table 11.4.

![Figure 11.19 Distortion versus rate characteristics for different transforms for a two-dimensional isotropic random field.](image-url)
Table 11.3 SNR Comparisons of Various Transform Coders for Random Fields with Isotropic Covariance Function, $\rho = 0.95$

<table>
<thead>
<tr>
<th>Block size</th>
<th>Rate bits/pixel</th>
<th>SNR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>KL</td>
<td>Cosine</td>
</tr>
<tr>
<td>8 x 8</td>
<td>0.25</td>
<td>11.74</td>
</tr>
<tr>
<td></td>
<td>0.50</td>
<td>13.82</td>
</tr>
<tr>
<td></td>
<td>1.00</td>
<td>16.24</td>
</tr>
<tr>
<td></td>
<td>2.00</td>
<td>20.95</td>
</tr>
<tr>
<td></td>
<td>4.00</td>
<td>31.61</td>
</tr>
<tr>
<td>16 x 16</td>
<td>0.25</td>
<td>12.35</td>
</tr>
<tr>
<td></td>
<td>0.50</td>
<td>14.2</td>
</tr>
<tr>
<td></td>
<td>1.00</td>
<td>16.58</td>
</tr>
<tr>
<td></td>
<td>2.00</td>
<td>21.26</td>
</tr>
<tr>
<td></td>
<td>4.00</td>
<td>31.90</td>
</tr>
</tbody>
</table>

Note: The KL transform would be nonseparable here.

Example 11.5 Choice of block size

The effect of block size on coder performance can easily be analyzed for the case of separable covariance random fields, that is, $r(m, n) = \rho |m| + |n|$. For a block size of $N \times N$, the covariance matrix can be written as

$$R = R \otimes R \Rightarrow |R|^2 = R^{2N} = (1 - \rho^2)^{2N(N-1)}$$

where $R$ is given by (2.68). Applying (11.69) the rate achievable by an $N \times N$ block KL transform coder is

$$R_{KL}(N) = \frac{1}{2} \log_2 \left( \frac{1 - \rho^2}{D} \right)^2 - \frac{1}{2N} \log_2 (1 - \rho^2)^2, \quad D = \left( \frac{1 - \rho^2}{1 + \rho} \right)^2$$

(11.75)

When plotted as a function of $N$ (Fig. 11.16), this shows the block size of $16 \times 16$ is suitable for $\rho = 0.95$. For higher values of the correlation parameter, $\rho$, the block size should be increased. Figure 11.20 shows some $16 \times 16$ block coded results.

Example 11.6 Choice of covariance model

The transform coefficient variances are important for designing the quantizers. Although the separable covariance model is convenient for analysis and design of transform coders, it is not very accurate. Figure 11.20 shows the results of $16 \times 16$ cosine transform coders based on the separable covariance model, the isotropic covariance model, and the actual measured transform coefficient variances. As expected, the actual measured variances yield the best coder performance. Generally, the isotropic covariance model performs better than the separable covariance model.

Zonal Versus Threshold Coding

Examination of bit allocation patterns (Fig. 11.18) reveals that only a small zone of transformed image is transmitted (unless the average rate was very high). Let $N_b$ be the number of transmitted samples. We define a zonal mask as the array

$$m(k, l) = \begin{cases} 1, & k, l \in I, \\ 0, & \text{otherwise} \end{cases}$$

(11.76)
which takes the unity value in the zone of largest \( N_1 \) variances of the transformed samples. Figure 11.21a shows a typical zonal mask. If we apply a zonal mask to the transformed blocks and encode only the nonzero elements, then the method is called \textit{zonal coding}.

In \textit{threshold coding} we encode the \( N_1 \) coefficients of largest amplitude rather than the \( N \), coefficients having the largest variances, as in zonal coding. The address set of transmitted samples is now

\[ I'_t = \{ k, l; |v(k, l)| > \eta \} \quad (11.77) \]

where \( \eta \) is a suitably chosen threshold that controls the achievable average bit rate. For a given ensemble of images, since the transform coefficient variances are fixed,
the zonal mask remains unchanged from one block to the next for a fixed bit rate. However, the threshold mask \( m_{\text{th}} \) (Fig. 11.21b) defined as

\[
m_{\text{th}}(k, l) = \begin{cases} 1, & k, l \in I'_i \\ 0, & \text{otherwise} \end{cases}
\]  

(11.78)
can change because \( I'_i \), the set of largest amplitude coefficients, need not be the same for different blocks. The samples retained are quantized by a suitable uniform quantizer followed by an entropy coder.

For the same number of transmitted samples (or quantizing bits), the threshold mask gives a better choice of transmission samples (that is, lower distortion). However, it also results in an increased rate because the addresses of the transmitted samples, that is, the boundary of the threshold mask, has to be coded for every image block. One method is to run-length code the transition boundaries in the threshold mask line by line. Alternatively, the two-dimensional transform coefficients are mapped into a one-dimensional sequence arranged in a predetermined order, such as in Fig. 11.21c. The thresholded sequence transitions are then run-length coded. Threshold coding is adaptive in nature and is useful for achieving high compression ratios when the image contents change considerably from block to block so that a fixed zonal mask would be inefficient.

**Fast KL Transform Coding**

For first-order AR sequences and for certain random fields represented by low-order noncausal models, fast KL transform coding approaches or exceeds the data compression efficiency of block KL transform coders. Recall from Section 6.5 that an \( N \times 1 \) vector \( u \) whose elements \( u(n), 1 \leq n \leq N \), come from a first-order, stationary, AR sequence with zero mean and correlation \( \rho \) has the decomposition

\[
u = u^0 + u^b
\]  

(11.79)

where \( u^b \) is completely determined by the boundary variables \( u(0) \) and \( u(N + 1) \) (see Fig. 6.8) and \( u^0 \) and \( u^b \) are mutually orthogonal random vectors. The KL transform of the sequence \( \{u^0(n), 1 \leq n \leq N\} \) is the sine transform, which is a fast transform. Thus (11.79) expresses the \( N \times 1 \) segment of a stationary Markov process as a two-source model. The first source has a fast KL transform, and the second source, has only two degrees of freedom (that is, it is determined by two variables).

Suppose we are given the \( N + 2 \) elements \( u(n), 0 \leq n \leq N + 1 \). Then the \( N \times 1 \) sequences \( u^0(n) \) and \( u^b(n) \) are realized as follows. First the boundary variables \( u(0) \) and \( u(N + 1) \) are passed through an interpolating FIR filter, which gives \( u^b(n) \), the best mean square estimate of \( u(n), 1 \leq n \leq N \), as

\[
u^b(n) = \alpha [Q^{-1}]_{n, 1} u(0) + \alpha [Q^{-1}]_{n, N} u(N + 1), \quad 1 \leq n \leq N
\]  

(11.80)

Then, we obtain the residual sequence

\[
u^0(n) \triangleq u(n) - u^b(n), \quad i \leq n \leq N
\]  

(11.81)

Instead of transform coding the original \( (N + 2) \times 1 \) sequence by its KL transform, \( u^0 \) and \( u^b \) can be coded separately using three different methods [6, 27]. One of these methods, called recursive block coding, is discussed here.
**Recursive block coding.** In the conventional block transform coding methods, the successive blocks of data are processed independently. The block size should be large enough so that interblock redundancy is minimal. But large block sizes spell large hardware complexity for the transformer. Also, at low bit rates (less than 1 bit per pixel) the block boundaries start becoming visibly objectionable.

\[ u = \{u(1), u(N)\} \]

![Diagram](image)

Figure 11.22 Fast KL transform coding (recursive block coding). Each successive block brings \((N + 1)\) new pixels.
In **recursive block coding** (RBC), the correlation between successive blocks of data is exploited through the use of block boundaries. This yields additional compression and allows the use of smaller block sizes (8 × 8 or less) without sacrificing performance. Moreover, this method significantly reduces the block boundary effects.

Figure 11.22 shows this method. For each block the boundary variable \( u(N + 1) \) is first coded. The reproduced value \( u'(N + 1) \) together with the initial value \( u'(0) \) (which is the boundary value of the previous block) are used to generate \( u^b(n) \), an estimate of \( u^b(n) \). The difference

\[
\tilde{u}^0(n) \triangleq u(n) - u^b(n)
\]

is sine transform coded. This yields better performance than the conventional block KLT coding (see Fig. 11.23).

**Remarks**

The FIR filter \( aQ^{-1} \) can be shown to be approximately the simple straight-line interpolator when \( \rho \) is close to 1 (see Problem 6.13 and [27]). Hence, \( u^b \) can be viewed as a *low-resolution copy* obtained by subsampling and interpolating the original image. This fact can be utilized in image archival applications, where only the low resolution image is retrieved in *search mode* and the residual image is called once the desired image has been located. This way the search process can be speeded up.
The foregoing theory can also be extended to second-order and higher AR models [27]. In these cases the KL transform of the residuals $u^0(n)$ is no longer a fast transform. This may not be a disadvantage for the recursive block coding algorithm because the transform size can now be quite small, so that a fast transform is not necessary.

In two dimensions many noncausal random field models yield fast KLT decompositions (see Example 6.17). Two-dimensional fast KLT coding algorithms similar to the ones just discussed can be designed using these decompositions. Figure 11.24 shows a two-dimensional recursive block coder. Figure 11.25 compares results of recursive block coding with cosine transform coding. The error images show the reduction in the block effects.

**Two-Source Coding**

The decomposition of (11.79) represents a stationary source by two sources, which can be realized from the image (block) and its boundary values. We could extend
this idea to represent an image as a nonstationary source

$$U = U_s + U_f$$

(11.82)

where $U_s$ is a stationary random field and $U_f$ is a deterministic component that represents certain features in the image. The two components are coded separately to preserve the different features in the image. One method, considered in [28], separates the image into its low- and high-spatial-frequency components. The high-frequency component, obtained by employing the Laplacian operator, is used to detect the edges whose locations are encoded. The low-frequency component can easily be encoded by transform or DPCM techniques. At the receiver a quantity
proportional to the Laplacian of a ramp function, called synthetic highs, is generated at the location of the edges. The reproduced image is the sum of the low-frequency component and the synthetic highs (Fig. 11.26a).

In other two source-coding schemes (Yan and Sakrison in [1c]), the stationary source is found by subtracting from the image a local average found by piecewise fitting planes or low-order surfaces through boundary or corner points. The corner points and changes in amplitude are coded (Fig. 11.26b). The residual signal, which is a much better candidate for a stationary random field model, is transform coded.

**Transform Coding Under Visual Criteria** [30]

From Section 3.3 we know that a weighted mean square criterion can be useful for visual evaluation of images. An FFT coder that incorporates this criterion (Fig. 11.27) quantizes the transform coefficients of the image contrast field weighted by $H(k, l)$, the sampled frequency response function of the visual system. Inverse weighting followed by the inverse FFT gives the reconstructed contrast field. To apply this method to block image coding, using arbitrary transforms, the image contrast field should first be convolved with $h(m, n)$, the sampled Fourier inverse of $H(\xi_1, \xi_2)$. The resulting field can then be coded by any desired method. At the receiver, the decoded field must then be convolved with the inverse filter whose frequency response is $1/H(\xi_1, \xi_2)$.

**Adaptive Transform Coding**

There are essentially three types of adaptation for transform coding:

1. Adaptation of transform
2. Adaptation of bit allocation
3. Adaptation of quantizer levels
Adaptation of the transform basis vectors is most expensive because a new set of KL basis vectors is required whenever any change occurs in the statistical parameters. A more practical method is to adapt the bit assignment of an image block, classified into one of several predetermined categories, according to the spatial activity (for instance, the variance of the data) in that block [1(c), p. 1285]. This results in a variable average rate from block to block but gives a better utilization of the total bits over the entire ensemble of image blocks. Another adaptive scheme is to allocate bits to image blocks so that each block has the same distortion [29]. This results in a uniform degradation of the image and appears less objectionable to the eye.

In adaptive quantization schemes, the bit allocation is kept constant but the quantizer levels are adjusted according to changes in the variances of the transform coefficients. Transform domain variances may be estimated either by updating the statistical parameters of the covariance model or by local averaging of the squared magnitude of the transform domain samples. Examples of adaptive transform coding are given in Section 11.7, where we consider interframe transform coding.

### Summary of Transform Coding

In summary, transform coding achieves relatively larger compression than predictive methods. Any distortion due to quantization and channel errors gets distributed, during inverse transformation, over the entire image (Fig. 11.40). Visually, this is less objectionable than predictive coding errors, which appear locally at the source. Although transform and predictive coding schemes are theoretically close in performance at low distortion levels for one-dimensional Markov sequences, their performance difference in two dimensions is substantial. This is because of two reasons. First, predictive coding is quite sensitive to changes in the statistics of the data. Therefore, in practice, only adaptive predictive coding schemes achieve the efficiency of (nonadaptive) transform coding methods. Second, in two dimensions,
finite-order causal predictors may never achieve compression ability close to transform coding because a finite-order causal representation of a two-dimensional random field may not exist. From an implementation point of view, predictive coding has much lower complexity both in terms of memory requirements and the number of operations to be performed. However, with the rapidly decreasing cost of digital hardware and computer memory, the hardware complexity of transform coders will not remain a disadvantage for very long. Table 11.4 summarizes the

<table>
<thead>
<tr>
<th>Design variables</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Covariance model</td>
<td>Convenient to analyze. Actual performance lower compared to nonseparable exponential.</td>
</tr>
<tr>
<td>Separable see (2.84)</td>
<td>Works well when ( \sigma^2, \alpha_1, \alpha_2 ) are matched to the image class.</td>
</tr>
<tr>
<td>Nonseparable see (2.85)</td>
<td>Useful in designing 2-D fast KLT coders [6].</td>
</tr>
<tr>
<td>Noncausal NC2 (see Section 6.9)</td>
<td>Choice depends on available memory size and the value of the one-step correlation ( \rho ). For ( \rho \leq 0.9 ), 16 x 16 size is adequate. A rule of thumb is to pick ( N ) such that ( \rho^N &lt;&lt; 1 ) (say 0.2). For smaller block size, recursive block coding is helpful.</td>
</tr>
<tr>
<td>2. Block size ( (N) )</td>
<td>This choice is important if block size is small, say ( N \leq 64 ).</td>
</tr>
<tr>
<td>16 x 16 to 64 x 64</td>
<td>Performs best for highly correlated data ( (\rho \geq 0.5) ).</td>
</tr>
<tr>
<td>3. Transform</td>
<td>Requires working with complex variables.</td>
</tr>
<tr>
<td>Cosine</td>
<td>Performs best for highly correlated data ( (\rho \geq 0.5) ).</td>
</tr>
<tr>
<td>Sine</td>
<td>For fast KL or recursive block coders.</td>
</tr>
<tr>
<td>DFT</td>
<td>Requires working with complex variables. Recommended if use of frequency domain is mandatory, such as in visual coding, and in CT, MRI image data, where source data has to pass through the Fourier domain.</td>
</tr>
<tr>
<td>Hadamard</td>
<td>Useful for small block sizes ( (= 4 \times 4) ). Implementation is much simpler than sinusoidal fast transforms.</td>
</tr>
<tr>
<td>Haar</td>
<td>Useful if higher spatial frequencies are to be emphasized. Poor compression on mean square basis.</td>
</tr>
<tr>
<td>KL</td>
<td>Optimum on mean square basis. Difficult to implement. Cosine or other sinusoidal transforms are preferable.</td>
</tr>
<tr>
<td>Slant</td>
<td>Best performance among nonsinusoidal fast transforms.</td>
</tr>
<tr>
<td>4. Quantizer</td>
<td>Either Laplacian or Gaussian density may be used. For dc coefficient a Rayleigh density may be used.</td>
</tr>
<tr>
<td>Lloyd-Max</td>
<td>Useful if the output is entropy coded or if the number of quantization levels is very large.</td>
</tr>
<tr>
<td>Optimum uniform</td>
<td></td>
</tr>
</tbody>
</table>
practical considerations in developing a design for transform coders. Table 11.5 compares the various transform coding schemes in terms of their compression ability. Here the compression ratio is the ratio of the number of bits per pixel in the original digital image (typically, 8) and the average number of bits per pixel required in encoding. Compression ratio values listed are to achieve SNR’s in the 30- to 36-dB range.

### 11.6 HYBRID CODING AND VECTOR DPCM

#### Basic Idea

The predictive coding techniques of Section 11.3 are based on raster scanning and scalar recursive prediction rules. If the image is vector scanned, for instance, a column at a time, then it is possible to generalize the DPCM techniques by considering vector recursive predictors. Hybrid coding is a method of implementing an $N \times 1$ vector DPCM coder by $N$ decoupled scalar DPCM coders. This is achieved by combining transform and predictive coding techniques. Typically, the image is unitarily transformed in one of its dimensions to decorrelate the pixels in that direction. Each transform coefficient is then sequentially coded in the other direction by one-dimensional DPCM (Fig. 11.28). This technique combines the advantages of hardware simplicity of DPCM and the robust performance of transform coding. The hardware complexity of this method is that of a one-dimensional transform coder and at most $N$ DPCM channels. In practice the number of DPCM

---

**Figure 11.28** Hybrid coding method.
channels is significantly less than \( N \) because many elements of the transformed vector are allocated zero bits and are therefore not coded at all.

**Hybrid Coding Algorithm.** Let \( u_n, n = 0, 1, \ldots, \) denote \( N \times 1 \) columns of an image, which are transformed as

\[
v_n = \Psi u_n, \quad n = 0, 1, 2, \ldots \tag{11.83}
\]

For each \( k \), the sequence \( v_n(k) \) is usually modeled by a first-order AR process [32], as

\[
v_n(k) = a(k)v_{n-1}(k) + b(k)e_n(k), \quad 1 \leq k \leq N, \ n \geq 0 \tag{11.84}
\]

\[
E[e_n(k)e_n(k')] = \sigma_e^2(k)\delta(k-k')\delta(n-n')
\]

The parameters of this model can be identified from the covariances of \( v_n(k), n = 0, 1, \ldots \), for each \( k \) (see Section 6.4). Some semicausal representations of images can also be reduced to such models (see Section 6.9). The DPCM equations for the \( k \)th channel can now be written as

**Predictor:** \( \bar{v}_n(k) = a(k)v_{n-1}(k) \)

**Quantizer input:** \( \bar{e}_n(k) = \frac{v_n(k) - \bar{v}_n(k)}{b(k)} \)

**Quantizer output:** \( \bar{e}_n(k) \)

**Filter:** \( v^*_n(k) = \bar{v}_n(k) + b(k)\bar{e}_n(k) \)

The receiver simply reconstructs the transformed vectors according to (11.85c) and performs the inverse transformation \( \Psi^{-1} \). Ideally, the transform \( \Psi \) should be the KL transform of \( u_n \). In practice, a fast sinusoidal transform such as the cosine or sine is used.

To complete the design we now need to specify the quantizer in each DPCM channel. Let \( B \) denote the average desired bit rate in bits per pixel, \( n_k \) be the number of bits allocated to the \( k \)th DPCM channel, and \( \sigma_e^2(k) \) be the quantizer mean square error in the \( k \)th channel, that is,

\[
B = \frac{1}{N} \sum_{k=1}^{N} n_k, \quad n_k \geq 0 \tag{11.86}
\]

Assuming that all the DPCM channels are in their steady state, the average mean square distortion in the coding of any vector (for noiseless channels) is simply the average of distortions in the various DPCM channels, that is,

\[
D = \frac{1}{N} \sum_{k=1}^{N} g_k(n_k)\sigma_e^2(k), \quad g_k(x) = \frac{f(x)}{1 - |a(k)|^2 f(x)} \tag{11.87}
\]

where \( f(x) \) and \( g_k(x) \) are the distortion-rate functions of the quantizer and the \( k \)th DPCM channel, respectively, for unity variance prediction error (see Problem 11.6). The bit allocation problem for hybrid coding is to minimize (11.87) subject to (11.86). This is now in the framework of the problem defined in Section 11.4, and the algorithms given there can be applied.
Example 11.7
Suppose the semicausal model (see section 6.9, and Eq. (6.106))
\[ u(m, n) = \alpha[u(m - 1, n) + u(m + 1, n)] + \gamma u(m, n - 1) + \epsilon(m, n), \]
\[ u(m, 0) = 0, \quad \forall m \]
\[ E[\epsilon(m, n)] = 0, \quad E[\epsilon(m, n)\epsilon(i, j)] = \beta^2 \delta(m - i, n - j) \]
is used to represent an \( N \times M \) image with high interpixel correlation. At the boundaries we can assume \( u(0, n) = u(1, n) \) and \( u(N, n) = u(N + 1, n) \). With these boundary conditions, this model has the realization of (11.84) for cosine transformed columns of the image with \( a(k) \triangleq \gamma / \lambda(k), b(k) \triangleq 1 / \lambda(k), \sigma_\epsilon^2(k) = \beta^2, \lambda(k) \triangleq 1 - 2 \alpha \cos (k - 1) \pi / N, \]
\[ 1 \leq k \leq N. \]
In an actual experiment, a \( 256 \times 256 \) image was coded in blocks of \( 16 \times 256 \).
The integer bit allocation for \( B = 1 \) was obtained as
\[ 3, 3, 3, 2, 2, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0 \]
Thus, only the first eight cosine transform coefficients of each \( 16 \times 1 \) column are used for DPCM coding. Figure 11.29a shows the result of hybrid coding using this model.

Adaptive Hybrid Coding
By updating the AR model parameters with variations in image statistics, adaptive hybrid coding algorithms can be designed [32]. One simple method is to apply the adaptive variance estimation algorithm [see Fig. 11.14a and eq. (11.38)] to each DPCM channel. The adaptive classification method discussed earlier gives better results especially at low rates. Each image column is classified into one of \( I \) (which is typically 4) predetermined classes that are fixed according to the variance distribution of the columns. Bits are allocated among different classes so that columns of high dynamic activity are assigned more bits than those of low activity. The class

![Figure 11.29 Hybrid encoded images at 1 bit/pixel. (a) Nonadaptive; (b) adaptive classification.](image-url)
membership information requires additional overhead of \( \log_2 I \) bits per column or \((1/N) \log_2 I\) bits per pixel. Figure 11.29 shows the result of applying adaptive algorithms to each DPCM channel of Example 11.7. Generally, the adaptive hybrid coding schemes can improve upon the nonadaptive techniques by about 3 dB, which is significant at low rates.

**Hybrid Coding Conclusions**

Hybrid coders combine the advantages of simple hardware complexity of DPCM coders and the high performance of transform coders, particularly at moderate bit rates (around 1 bit per pixel). Its performance lies between transform coding and DPCM. It is easily adaptable to noisy images [6, 32] and to changes in image statistics. It is particularly useful for interframe image data compression of motion images, as we shall see in Section 11.7. It is less sensitive to channel errors than DPCM but is not as robust as transform coding. Hybrid coders have been implemented for real-time data compression of images acquired by remotely piloted vehicles (RPV) [33].

**11.7 INTERFRAME CODING**

Teleconferencing, broadcast, and many medical images are received as sequences of two-dimensional image frames. Interframe coding techniques exploit the redundancy between the successive frames. The differences between successive frames are due to object motion or camera motion, panning, zooming, and the like.

**Frame Repetition**

Beyond the horizontal and/or vertical line interlace methods discussed in Section 11.1, a simple method of interframe compression is to subsample and frame-repeat interlaced pictures. This, however, does not produce good-quality moving images. An alternative is selective replenishment, where the frames are transmitted at a reduced rate according to a fixed, predetermined updating algorithm. At the receiver, any nonupdated data is refreshed from the previous frame stored in the frame memory. This method is reasonable for slow-moving areas only.

**Resolution Exchange**

The response of the human visual system is poor for dynamic scenes that simultaneously contain high spatial and temporal frequencies. Thus, rapidly changing areas of a scene can be represented with reduced amplitude and spatial resolution when compared with the stationary areas. This allows exchange of spatial resolution with temporal resolution and can be used to produce good-quality images at data rates of 2–2.5 bits per pixel. One such method segments the image into stationary and moving areas by thresholding the value of the frame-difference signal. In stationary areas frame differences are transmitted for every other pixel and the remaining
pixels are repeated from the previous frame. In moving areas 2:1 horizontal sub-sampling is used, with intervening elements restored by interpolation along the scan lines. Using 5-bits-per-pixel frame-differential coding, a channel rate of 2.5 bits per pixel can be achieved. The main distortion occurs at sharp edges moving with moderate speed.

**Conditional Replenishment**

This technique is based on detection and coding of the moving areas, which are replenished from frame to frame. Let \( u(m, n, i) \) denote the pixel at location \((m, n)\) in frame \(i\). The interframe difference signal is

\[
e(m, n, i) = u(m, n, i) - u'(m, n, i - 1)
\]

where \( u'(m, n, i - 1) \) is the reproduced value of \( u(m, n, i - 1) \) in the \((i - 1)\)st frame. Whenever the magnitude of \( e(m, n, i) \) exceeds a threshold \( \eta \), it is quantized and coded for transmission. At the receiver, a pixel is reconstructed either by repeating the value of that pixel location from the previous frame if it came from a stationary area, or it is replenished by the decoded difference signal if it came from a moving area, giving

\[
u'(m, n, i) = \begin{cases} u'(m, n, i - 1) + e'(m, n, i), & \text{if } |e(m, n, i)| > \eta \\ u'(m, n, i - 1), & \text{otherwise} \end{cases}
\]

For transmission, code words representing the quantized values and their addresses are generated. Isolated points or very small clusters of moving areas are ignored to make the address coding scheme efficient. A reasonable-size buffer with appropriate buffer-control strategy is necessary to achieve a steady bit rate. With insufficient buffer size, its control can require extreme action (such as stopping the coder temporarily), which can cause jerky reproduction of motion (Fig. 11.30a). Simulation studies [6, 39] have shown that with a suitably large buffer a 1-bit-per-pixel rate can be achieved conveniently with an average SNR' of about 34 dB (39 dB in stationary areas and 30 dB in moving areas). Figure 11.30b shows an encoded image and the encoding error magnitudes for a typical frame.

**Adaptive Predictive Coding**

Adaptations to motion characteristics can yield considerable gains in performance of interframe predictive coding methods. Figure 11.31 shows one such method, where the incoming pixel is classified as belonging to an area of stationary \((C_s)\), moderate/slow \((C_M)\), or rapid \((C_R)\) motion. Classification is based on an activity index \( \alpha(m, n, i) \), which is the absolute sum of interframe differences of a neighborhood \( \mathcal{N} \) (Fig. 11.11) of previously encoded pixels, that is,

\[
\alpha(m, n, i) = \sum_{(x, y) \in \mathcal{N}} |u'(m + x, n + y, i) - u'(m + x, n + y, i - 1)|
\]

\[
\mathcal{N} = \{(0, -s), (-1, -1), (-1, 0), (-1, 1)\}
\]
(a) Frame Replenishment Coding.
Bit-rate = .5 bit/pixel, SNR' = 28.23 dB.

(b) Frame Replenishment Coding.
Bit-rate = 1 bit/pixel, SNR' = 34.19 dB.

(c) Adaptive Classification Prediction Coding.
Bit-rate = .5 bit/pixel, SNR' = 34.35 dB.

Figure 11.30  Results of interframe predictive schemes.
where $s = 2$ in 2:1 subsampling mode and is unity otherwise. A large value of $\alpha(m, n, i)$ indicates large motion in the neighborhood of the pixel. The predicted value of the current pixel, $u(m, n, i)$, is

$$
\bar{u}(m, n, i) = \begin{cases} 
    u^*(m, n, i - 1), & (m, n) \in C_S \\
    u^*(m - p, n - q, i - 1), & (m, n) \in C_M \\
    \rho_1 u^*(m, n - 1, i) + \rho_2 u^*(m - 1, n, i) \\
    - \rho_1 \rho_2 u^*(m - 1, n - 1, i), & (m, n) \in C_R 
\end{cases}
$$

(11.92)

where $\rho_1$ and $\rho_2$ are the one-pixel correlations coefficients along $m$ and $n$, respectively. Displacements $p$ and $q$ are chosen by estimating the average displacement of the neighborhood that gives the minimum activity. Observe that for the case of rapid motion, the two-dimensional predictor of (11.35a) is used. This is because temporal prediction would be difficult for rapidly changing areas. The number of quantizer levels used for each class is proportional to its activity. This method achieves additional compression by a factor of two over the conditional replenishment while maintaining approximately the same SNR (Fig. 11.30c). For greater details on coder design, see [39].

**Predictive Coding with Motion Compensation**

In principle, if the motion trajectory of each pixel could be measured, then only the initial frame and the trajectory information would need to be coded. To reproduce the images we could simply propagate each pixel along its trajectory. In practice, the motion of objects in the scene can be approximated by piecewise displacements from frame to frame. The displacement vector is used to direct the

---

Figure 11.31 Interframe adaptive predictive coding.
motion-compensated interframe predictor. The success of a motion-compensated coder depends on accuracy, speed, and robustness (with respect to noise) of the displacement estimator.

**Displacement Estimation Algorithms**

1. **Search techniques.** Search techniques look for a displacement vector \( \mathbf{d} \triangleq [p, q]^T \) such that a distortion function \( D(p, q) \) between a reference frame and the current frame is minimized. Examples are template matching, logarithmic search, hierarchical search, conjugate direction, and gradient search techniques, which are discussed in Section 9.12. In these techniques the log search converges most rapidly when the search area is large. For small search areas, the conjugate direction search method is simpler. These techniques are quite robust and are useful especially when the displacement is constant for a block of pixels. For interframe motion estimation the search can be usually limited to a window of \( 5 \times 5 \) pixels.

2. **Recursive displacement estimation.** To understand this algorithm it is convenient to consider the continuous function \( u(x, y, t) \) representing the image frame at time \( t \). Given a displacement error measure \( f(x) \), one possibility is to update \( \mathbf{d} \) recursively via the gradient algorithm [37]

\[
\mathbf{d}_k = \mathbf{d}_{k-1} - \varepsilon \nabla \mathbf{d}_k \mathbf{f}_{k-1}
\]

\[
f_k \triangleq f[u(x, y, t) - u(x - p_k, y - q_k, t - \tau)]
\]

where \( \mathbf{d}_k \) is the displacement estimate at pixel \( k \) for the adopted scanning sequence, \( \nabla \mathbf{d} \) is the gradient of \( f \) with respect to \( \mathbf{d} \), and \( \tau \) is the interframe interval. The \( \varepsilon \) is a small positive quantity that controls the correction at each recursion. For \( f(x) = x^2/2 \), (11.93) reduces to

\[
\mathbf{d}_k = \mathbf{d}_{k-1} - \varepsilon [u(x, y, t) - u(x - p_{k-1}, y - q_{k-1}, t - \tau)] \cdot 
\left[ \begin{array}{c} 
\frac{\partial u}{\partial x} (x - p_{k-1}, y - q_{k-1}, t - \tau) \\
\frac{\partial u}{\partial y} (x - p_{k-1}, y - q_{k-1}, t - \tau)
\end{array} \right]
\]

Since \( u(x, y, t) \) is available only at sampling locations, interpolation is required to evaluate \( u, \partial u/\partial x, \) and \( \partial u/\partial y \) at \( (x - p_k, y - q_k, t - \tau) \). The advantage of this algorithm is that it estimates displacement values for each pixel. However, it lacks the robustness of block search techniques.

3. **Differential techniques.** These algorithms are based on the fact that the gray-level value of a pixel remains constant along its path of motion, that is,

\[
u(x(t), y(t), t) = \text{constant}
\]

where \( x(t), y(t) \) is the motion trajectory. Differentiating both sides with respect to \( t \), we obtain

\[
\frac{\partial u}{\partial t} + v_1 \frac{\partial u}{\partial x} + v_2 \frac{\partial u}{\partial y} = 0, \quad (x, y) \in \mathcal{R}
\]
where \( v_1 \triangleq \frac{dx(t)}{dt}, \) \( v_2 \triangleq \frac{dy(t)}{dt}, \) are the two velocity components and \( \mathcal{S} \) is the region of moving pixels having the same motion trajectory. The displacement vector \( \mathbf{d} \) can be estimated as

\[
d = \int_{t_0}^{t_+} \mathbf{v} \, dt = \mathbf{v}_r
\]  

(11.97)

assuming the velocity remains constant during the frame intervals. The velocity vector can be estimated from the interframe data after it has been segmented into stationary and moving areas [41] by minimizing the function

\[
J \triangleq \iint_{\mathcal{S}} \left[ \frac{\partial u}{\partial t} + v_1 \frac{\partial u}{\partial x} + v_2 \frac{\partial u}{\partial y} \right]^2 \, dx \, dy
\]  

(11.98)

Setting \( \nabla J = 0 \) gives the solution as

\[
\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} c_{xx} & c_{xy} \\ c_{yx} & c_{yy} \end{bmatrix}^{-1} \begin{bmatrix} c_{xt} \\ c_{yt} \end{bmatrix}
\]

(11.99)

where \( c_{\alpha\beta} \) denotes the correlation between \( \frac{\partial u}{\partial \alpha} \) and \( \frac{\partial u}{\partial \beta} \), that is,

\[
c_{\alpha\beta} \triangleq \iint_{\mathcal{S}} \frac{\partial u}{\partial \alpha} \frac{\partial u}{\partial \beta} \, dx \, dy \quad \text{for} \ \alpha, \beta = x, y, t
\]  

(11.100)

This calculation can be speeded up by estimating the correlations as [40]

\[
c_{\alpha\beta} = \iint_{\mathcal{S}} \frac{\partial u}{\partial \alpha} \text{sgn} \left( \frac{\partial u}{\partial \beta} \right) \, dx \, dy
\]  

(11.101)

Thus \( \mathbf{v} \) can be calculated from the partial differentials of \( u(x, y, t) \), which can be approximated from the given interframe sampled data. Like the block search algorithms, the differential techniques also give the displacement vector for a block or a region. These methods are faster, although still not as robust as the block search algorithms.

Having estimated the motion, compression is achieved by skipping image frames and reproducing the missing frames at the receiver either by frame repetition or by interpolation along the motion trajectory. For example, if the alternate frames, say \( u(m, n, 2i), \) \( i = 1, 2, \ldots \), have been skipped, then with motion compensation we have

Frame repetition: \( u'(m, n, 2i) = u'(m - p, n - q, 2i - 1) \)  

(11.102)

Frame interpolation: \( u'(m, n, 2i) = \frac{1}{2} [u'(m - p, n - q, 2i - 1) + u'(m - p', n - q', 2i + 1)] \)  

(11.103)

where \( (p, q) \) and \( (p', q') \) are the displacement vectors relative to the preceding and following frames, respectively. Without motion compensation, we would set \( p = q = p' = q' = 0 \). Figure 11.32 shows the advantage of motion compensation in frame skipping. The improvement due to motion compensation, roughly 10 dB, is quite significant.
Interframe Hybrid Coding

Hybrid coding is particularly useful for interframe image data compression of motion images. A two-dimensional $M \times N$ block of the $i$th frame, denoted by $U_i$, is first transformed to give $V_i$. For each $(k, l)$ the sequence $v_i(k, l), i = 1, 2, \ldots$, is considered a one-dimensional random process and is coded independently by a suitable one-dimensional DPCM method. The receiver reconstructs $v_i(k, l)$ and

Along temporal axis, $\text{SNR}' = 16.90$ dB

Along motion trajectory, $\text{SNR}' = 26.69$ dB

(a) Frame repetition (or interframe prediction) based on the preceding frame

Figure 11.32 Effects of motion compensation on interframe prediction and interpolation.
Along temporal axis, SNR’ = 19.34 dB

Along motion trajectory, SNR’ = 29.56 dB

(b) Frame interpolation from the preceding and the following frames

Figure 11.32 (Cont’d)

performs its two-dimensional inverse transform. A typical method uses the discrete cosine transform and a first-order AR model for each DPCM channel. In a motion-compensated hybrid coder the DPCM prediction error becomes

\[ e_{i}(k, l) = v_{i}(k, l) - \alpha \tilde{v}_{i-1}(k, l) \]  

(11.104)

where \( \tilde{v}_{i-1}(k, l) \) are obtained by transforming the motion-compensated sequence \( u_{i-1}(m - p, n - q) \). If \( \alpha \), the prediction coefficient, is constant for each channel \( (k, l) \), then \( e_{i}(k, l) \) would be the same as the transform of \( u_{i}(m, n) - \alpha u_{i-1}(m - p, n - q) \). This yields a motion-compensated hybrid coder, as shown in
Fig. 11.33. Results of different interframe hybrid coding methods are shown in Fig. 11.34. These and other results [6, 36] show that with motion compensation, the adaptive hybrid coding method performs better than adaptive predictive coding and adaptive three-dimensional transform coding. However, the coder now requires two sets of two-dimensional transformations.

Three-Dimensional Transform Coding

In many applications, (for example, in multispectral imaging, interframe video imaging, medical cineangiography, CT scanning, and so on), we have to work with three- (or higher-) dimensional data. Transform coding schemes are possible for compression of such data by extending the basic ideas of Section 11.5. A three-dimensional (separable) transform of a $M \times N \times I$ sequence $u(m, n, i)$ is defined as

$$v(k, l, j) \triangleq \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \sum_{i=0}^{I-1} u(m, n, i) a_M(k, m) a_N(l, n) a_I(j, i)$$

(11.105)

where $0 \leq (k, m) \leq M - 1$, $0 \leq (l, n) \leq N - 1$, $0 \leq (j, i) \leq I - 1$, and $\{a_M(k, m)\}$ are the elements of an $M \times M$ unitary matrix $A_M$, and so on. The transform coefficients given by (11.105) are simply the result of taking the $A$-transform with respect to each index and will require $MNI \log_2(MNI)$ operations for a fast transform. The storage requirement for the data is $MNI$. As before, the practical approach is to partition the data into small blocks (such as $16 \times 16 \times 16$) and process each block independently. The coding algorithm after transformation is the same as before except that we are working with triple indexed variables. Figure 11.35 shows results for one frame of a sequence of cosine transform coded images. The result of Fig. 11.35a corresponds to the use of the three-dimensional separable covariance model

$$r(m, n, i) = \sigma^2 \rho_1|^{m|} \rho_2|^{n|} \rho_3|^{l|}$$
Figure 11.34 Interframe hybrid coding
(a) 0.5 bit/pixel, separable covariance model, $\text{SNR}' = 32.1 \text{ dB}$;

(b) 0.5 bit/pixel, measured covariances, $\text{SNR}' = 36.8 \text{ dB}$;

(c) 0.5 bit/pixel, measured covariances, adaptive, $\text{SNR}' = 41.2 \text{ dB}$.

**Figure 11.35** Interframe transform coding
which, as expected, performs poorly. Also, the adaptive hybrid coding with motion compensation performs better than three-dimensional transform coding. This is because incorporating motion information in a three-dimensional transform coder requires selecting spatial blocks along the motion trajectory, which is not a very attractive alternative.

11.8 IMAGE CODING IN THE PRESENCE OF CHANNEL ERRORS

So far we have assumed the channel between the coder and the decoder to be noiseless. To account for channel errors, we have to add redundancy to the input by appending error correcting bits. Thus a proper trade-off between source coding (redundancy removal) and channel coding (redundancy injection) has to be achieved in the design of data compression systems. Often, the error-correcting codes are designed to reduce the probability of bit errors, and for simplicity, equal protection is provided to all the samples. For image data compression algorithms, this does not minimize the overall error. In this section we consider source-channel-encoding methods that minimize the overall mean square error.

Consider the PCM transmission system of Fig. 11.36, where a quantizer generates $k$-bit outputs $x_i \in S$, which are mapped, one-to-one, into $n$-bit ($n \geq k$) codewords $g_i \in C$. Let $\beta(\cdot)$ denote this mapping. The channel is assumed to be memoryless and binary symmetric with bit error probability $p_e$. It maps the set $C$ of $K = 2^k$ possible $n$-bit code words into a set $V$ of $2^n$ possible $n$-bit words. At the receiver, $\lambda(\cdot)$ denotes the mapping of elements of $V$ into the elements on the real line $R$. The identity element of $V$ is the vector $\mathbf{0} \triangleq [0, 0, \ldots, 0]$.

The Optimum Mean Square Decoder

The mean square error between the decoder output and the encoder input is given by

$$
\sigma_x^2 = \sigma_x^2(\beta, \lambda) \triangleq E[(y - x)^2] = \sum_{v \in V} p(v) \left( \sum_{x \in S} (\lambda(v) - x)^2 p(x|v) \right)
$$

and depends on the mappings $\beta(\cdot)$ and $\lambda(\cdot)$. From estimation theory (see Section 2.12) we know that given the encoding rule $\beta$, the decoder that minimizes this error is given by the conditional mean of $x$, that is,

$$
y = \lambda(v) = \sum_{x \in S} x p(x|v) = E[x|v]
$$

where $p(x|v)$ is the conditional density of $x$ given the channel output $v$. The function $\lambda(v)$ need not map the channel output into the set $S$ even if $n = k$.

![Figure 11.36 Channel coding for PCM transmission.](image-url)
The Optimum Encoding Rule

The optimum encoding rule $\beta(\cdot)$ that minimizes (11.106) requires an exhaustive search for the optimum subspace $C$ over all subspaces of $V$. Practical solutions are found by restricting the search to a particular set of subspaces [39, 42]. Table 11.6 shows one set of practical basis vectors for uniformly distributed sources, from

**TABLE 11.6 Basic Vectors $\{\phi_i, i = 1, \ldots, k\}$ for $(n, k)$ Group Codes**

<table>
<thead>
<tr>
<th>$i$</th>
<th>$n - k$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>1000000</td>
<td>1000000</td>
<td>10000000110</td>
<td>10000000110</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>01000000</td>
<td>0100000001</td>
<td>010000000110</td>
<td>010000000110</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>00100000</td>
<td>00100000001</td>
<td>001000000110</td>
<td>001000000110</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>00010000</td>
<td>00010000001</td>
<td>000100000110</td>
<td>000100000110</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>00001000</td>
<td>00001000001</td>
<td>000010000110</td>
<td>000010000110</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>00000100</td>
<td>00000100001</td>
<td>000001000110</td>
<td>000001000110</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>00000010</td>
<td>00000010001</td>
<td>000000100110</td>
<td>000000100110</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>00000001</td>
<td>00000001001</td>
<td>000000010110</td>
<td>000000010110</td>
<td></td>
</tr>
</tbody>
</table>

$n = 11, k = 6$

$i \rightarrow 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6$

$g_i \rightarrow 100000111101 \quad 01000010100 \quad 00100010111 \quad 00010001100 \quad 00001000001 \quad 000001000000$
which $\beta(\cdot)$ is obtained as follows. Let $b^{\Delta} = [b(1), b(2), \ldots, b(k)]$ be the binary representation of an element of $S$; then

$$g = \beta(b) = \sum_{i=1}^{k} b(i) \cdot \phi_i$$  \hspace{1cm} (11.108)

where $\Sigma \oplus$ denotes exclusive-OR summation and $\cdot$ denotes the binary product. The codes generated by this method are called the $(n, k)$ group codes.

**Example 11.8**

Let $n = 4$ and $k = 2$, so that $n - k = 2$. Then $\phi_1 = [1 \ 0 \ 1 \ 1]$, $\phi_2 = [0 \ 1 \ 0 \ 1]$, and $\beta(\cdot)$ is given as follows

<table>
<thead>
<tr>
<th>$x$</th>
<th>$b$</th>
<th>$g = \beta(b)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 0</td>
<td>0 = 0 \cdot \phi_1 \oplus 0 \cdot \phi_2</td>
</tr>
<tr>
<td>1</td>
<td>0 1</td>
<td>1 = 0 \cdot \phi_1 \oplus 1 \cdot \phi_2</td>
</tr>
<tr>
<td>2</td>
<td>1 0</td>
<td>1 = 1 \cdot \phi_1 \oplus 0 \cdot \phi_2</td>
</tr>
<tr>
<td>3</td>
<td>1 1</td>
<td>0 = \phi_1 \oplus \phi_2</td>
</tr>
</tbody>
</table>

In general the basis vectors $\phi_i$ depend on the bit error probability $p_e$ and the source probability distribution. For other distributions, Table 11.5 is found to lower the channel coding performance only slightly for $p_e \ll 1$ [39]. Therefore, these group codes are recommended for all mean square channel coding applications.

**Optimization of PCM Transmission**

If $\eta_c$ and $\eta_q$ denote the channel and the quantizer errors, we can write (from Fig. 11.36) the input sample as

$$z = x + \eta_q = y + \eta_c + \eta_q$$  \hspace{1cm} (11.109)

This gives the total mean square error as

$$\sigma_i^2 = E[(z - y)^2] = E[(\eta_c + \eta_q)^2]$$  \hspace{1cm} (11.110)

For a fixed channel coder $\beta(\cdot)$, this error is minimum when [6, 39]  

1. $\eta_c$ and $\eta_q$ are orthogonal
2. $\sigma_c^2 \Delta = E[\eta_c^2]$ and $\sigma_q^2 \Delta = E[\eta_q^2]$ are minimum.

This requires

$$y \Delta= \lambda(v) = E[x \mid v]$$  \hspace{1cm} (11.111)$$

$$x = \alpha(z) = E[z \mid z \in \mathcal{I}]$$  \hspace{1cm} (11.112)

where $\mathcal{I}_i, i = 1, \ldots, 2^k$ denotes the $i$th quantization interval of the quantizer.

This result says that the optimum decoder is independent of the optimum quantizer, which is the Lloyd-Max quantizer. Thus the overall optimal design can be accomplished by optimizing the quantizer and the decoder individually. This gives

$$\sigma_i^2 = \sigma_c^2 + \sigma_q^2$$  \hspace{1cm} (11.113)

Let $f(k)$ and $c(n, k)$ denote the mean square distortions due to the $k$-bit Lloyd-Max quantizer and the channel, respectively, when the quantizer input is a unit variance random variable (Tables 11.7 and 11.8). Then we can write the total
### Table 11.7 Quantizer Distortion Function $f(k)$ for Unity Variance Inputs

<table>
<thead>
<tr>
<th>Density</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaussian</td>
<td>0.3634</td>
<td>0.1175</td>
<td>0.0346</td>
<td>0.0095</td>
<td>$2.5 \times 10^{-3}$</td>
<td>$6.4 \times 10^{-4}$</td>
<td>$1.6 \times 10^{-4}$</td>
<td>$4 \times 10^{-5}$</td>
</tr>
<tr>
<td>Laplacian</td>
<td>0.3634</td>
<td>0.1762</td>
<td>0.0545</td>
<td>0.0154</td>
<td>$4.1 \times 10^{-3}$</td>
<td>$1.06 \times 10^{-3}$</td>
<td>$2.7 \times 10^{-4}$</td>
<td>$7 \times 10^{-5}$</td>
</tr>
<tr>
<td>Uniform</td>
<td>0.2500</td>
<td>0.0625</td>
<td>0.0156</td>
<td>0.0039</td>
<td>$9.77 \times 10^{-4}$</td>
<td>$2.44 \times 10^{-4}$</td>
<td>$6.1 \times 10^{-5}$</td>
<td>$1.52 \times 10^{-5}$</td>
</tr>
</tbody>
</table>

### Table 11.8 Channel Distortion $c(n, k)$ for $(n, k)$ Block Coding of Outputs of a Quantizer with Unity Variance Input

<table>
<thead>
<tr>
<th>Input Density</th>
<th>$n - k$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$k$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>0.0252</td>
<td>0.0129</td>
<td>0.0075</td>
<td>0.0004</td>
<td>0.0003</td>
<td>0.00254</td>
<td>0.0013</td>
<td>0.8 x 10^{-5}</td>
<td>0.4 x 10^{-6}</td>
<td>&lt;10^{-6}</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.0483</td>
<td>0.0280</td>
<td>0.0066</td>
<td>0.0018</td>
<td>0.0010</td>
<td>0.00505</td>
<td>0.0028</td>
<td>0.0005</td>
<td>0.00002</td>
<td>0.00001</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.0656</td>
<td>0.0372</td>
<td>0.0115</td>
<td>0.0033</td>
<td>0.0025</td>
<td>0.0069</td>
<td>0.0038</td>
<td>0.0010</td>
<td>0.00004</td>
<td>0.00003</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.0765</td>
<td>0.0423</td>
<td>0.0138</td>
<td>0.0056</td>
<td>0.0032</td>
<td>0.0083</td>
<td>0.0044</td>
<td>0.0012</td>
<td>0.00006</td>
<td>0.00003</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.0821</td>
<td>0.0450</td>
<td>0.0149</td>
<td>0.0062</td>
<td>0.0036</td>
<td>0.0093</td>
<td>0.0047</td>
<td>0.0013</td>
<td>0.00001</td>
<td>0.00006</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0.0856</td>
<td>0.0463</td>
<td>0.0154</td>
<td>0.0064</td>
<td>0.0037</td>
<td>0.0101</td>
<td>0.0049</td>
<td>0.0014</td>
<td>0.00008</td>
<td>0.00008</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0.0923</td>
<td>0.0477</td>
<td>0.0156</td>
<td>0.0068</td>
<td>0.0037</td>
<td>0.0112</td>
<td>0.0051</td>
<td>0.0014</td>
<td>0.00012</td>
<td>0.00008</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0.1050</td>
<td>0.0508</td>
<td>0.0169</td>
<td>0.0076</td>
<td></td>
<td>0.0143</td>
<td>0.0056</td>
<td>0.0015</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For $p_e = 0.01$:
- $10^{-4}$
- $10^{-5}$
- $10^{-6}$

For $p_e = 0.001$:
- $10^{-4}$
- $10^{-5}$
- $10^{-6}$
mean square error for the input of $z$ of variance $\sigma_z^2$ as

$$\begin{align*}
\sigma_i^2 &= \sigma_z^2 \hat{\sigma}_i^2, \\
\sigma_q^2 &\triangleq \sigma_z^2 f(k), \\
\sigma_c^2 &\triangleq \sigma_z^2 c(n, k)
\end{align*}$$

(11.114)

For a fixed $n$ and $k \leq n$, $f(k)$ is a monotonically decreasing function of $k$, whereas $c(n, k)$ is a monotonically increasing function of $k$. Hence for every $n$ there is an optimum value of $k = k_0(n)$ for which $\hat{\sigma}_i^2$ is minimized. Let $d(n)$ denote the minimum value of $\hat{\sigma}_i^2$ with respect to $k$, that is,

$$d(n) = \min_{k} \{\hat{\sigma}_i^2(n, k)\} \triangleq \hat{\sigma}_i^2(n, k_0(n))$$

(11.115)

Figure 11.37 shows the plot of the distortions $\hat{\sigma}_i^2(n, n)$ and $d(n)$ versus the rate for $n$-bit PCM transmission of a Gaussian random variable when $p_e = 0.01$. The quantity $\hat{\sigma}_i^2(n, n)$ represents the distortion of the PCM system if no channel error protection is provided and all the bits are used for quantization. It shows, for example, that optimum combination of error protection and quantization could improve the system performance by about 11 dB for an 8-bit transmission.

**Channel Error Effects in DPCM**

In DPCM the total error in the reconstructed image can be written as

$$\delta u(i, j) = \eta_q(i, j) + \sum_{i'} \sum_{j'} h(i - i', j - j') \eta_c(i', j')$$

(11.116)

where $\eta_q$ is now the DPCM quantizer noise and $h(i, j)$ is the impulse response of the reconstruction filter.

![Figure 11.37 Distortion versus rate characteristics of PCM transmission over a binary symmetric channel.](image)
It is essential that the reconstruction filter be stable to prevent the channel errors from accumulating to arbitrarily large values. Even when the predictor models are stable, the channel mean square error gets amplified by a factor $\sigma_i^2/\beta^2$ by the reconstruction filter where $\beta^2$ is the theoretical prediction error variance (without quantizer) (Problem 11.5). For the optimum mean square channel decoder the total mean square error in the reconstructed pixel at $(i, j)$ can be written as

$$\sigma_i^2 = \sigma_v^2 + \frac{\sigma_c^2 \sigma_u^2}{\beta^2} = \sigma_v^2 \left[f(k) + \frac{1}{\beta^2} c(n, k)\right]$$

(11.117)

where $\hat{\beta}^2 = \beta^2/\sigma_u^2$, and $\sigma_v^2$ is the variance of the actual prediction error in the DPCM loop. Recall that high compression is achieved for small values of $\hat{\beta}^2$. Equation (11.117) shows that the higher the compression, the larger is the channel error amplification. Visually, channel noise in DPCM tends to create two-dimensional patterns that originate at the channel error locations and propagate until the reconstruction filter impulse response decays to zero (see Fig. 11.38). In line-by-line DPCM, streaks of erroneous lines appear. In such cases, the erroneous line can be replaced by the previous line or by an average of neighboring lines. A median filter operating orthogonally to the scanning direction can also be effective.

To minimize channel error effects, $\sigma_i^2$ given by (11.117) must be minimized to find the quantizer optimum allocation $k = k(n)$ for a given overall rate of $n$ bits per pixel.

**Example 11.9**

A predictor with $a_1 = 0.848, a_2 = 0.755, a_3 = -0.608, a_4 = 0$ in (11.35a) and $\hat{\beta}^2 = 0.019$ is used for DPCM of images. Assuming a Gaussian distribution for the quantizer input, the optimum pairs $[n, k(n)]$ are found to be:

| TABLE 11.9 Optimum Pairs $[n, k(n)]$ for DPCM Transmission |
|-------------|-------------|-------------|-------------|-------------|
| $n$ | $k(n)$ | $p_e = 0.01$ | $p_e = 0.001$ |
| 1 | 1 | 2 | 3 | 4 | 5 | 1 | 2 | 3 | 4 | 5 |
| 2 | 1 | 2 | 2 | 2 | 2 | 3 |

This shows that if the error rate is high ($p_e = 0.01$), it is better to protect against channel errors than to worry about the quantizer errors. To obtain an optimum pair, we evaluate $\sigma_i^2/\sigma_v^2$ via (11.117) and Tables 11.6, 11.7, and 11.8 for different values of $k$ (for each $n$). Then the values of $k$ for which this quantity is minimum is found. For a given choice of $(n, k)$, the basis vectors from Table 11.6 can be used to generate the transmission code words.

**Optimization of Transform Coding**

Suppose a channel error causes a distortion $\delta v(k, l)$ of the $(k, l)$th transform coefficient. This error manifests itself by spreading in the reconstructed image in proportion to the $(k, l)$th basis image, as

$$\delta u(i, j) = \delta v(k, l) \phi_k(i) \phi_l(j)$$

(11.118)
Figure 11.38 Two bits/pixel DPCM coding in the presence of a transmission error rate of $10^{-3}$. (a) Propagation of transmission errors for different predictors. Clockwise from top left, error location: optimum, three point, and two point predictors. (b) Optimum linear predictor. (c) Two point predictor $\gamma \left( \frac{A + D}{2} \right)$. (d) Three-point predictor $\gamma (A + C - B)$.

This is actually an advantage of transform coding over DPCM because, for the same mean square value, localized errors tend to be more objectionable than distributed errors. The foregoing results can be applied for designing transform coders that protect against channel errors. A transform coder contains several PCM channels, each operating on one transform coefficient. If we represent $z_j$ as the $j$th transform coefficient with variance $\sigma_j^2$, then the average mean square distortion of a transform coding scheme in the presence of channel errors becomes

$$D = \sum_j \sigma_j^2 d(n_j)$$  \hspace{1cm} (11.119)

where $n_j$ is the number of bits allocated to the $j$th PCM channel. The bit allocation algorithm for a transform coder will now use the function $d(n)$, which can be
Figure 11.39 Bit allocations for quantizers and channel protection in $16 \times 16$ block, cosine transform coding of images modeled by the isotropic covariance function. Average bit rate is 1 bit/pixel, and channel error is 1%.

Figure 11.40 Transform coding in the presence of channel errors.
evaluated via (11.115). Knowing $n_j$, we can find $k_j = k(n_j)$, the corresponding optimum number of quantizer bits.

Figure 11.39 shows the bit allocation pattern $k_0(i, j)$ for the quantizers and the allocation of channel protection bits $(n(i, j) - k_0(i, j))$ at an overall average bit rate of 1 bit per pixel for $16 \times 16$ block coding of images modeled by the isotropic covariance function. As expected, more protection is provided to samples that have larger variances (and are, therefore, more important for transmission). The overhead due to channel protection, even for the large value of $p_e = 0.01$, is only 15%. For $p_e = 0.001$, the overhead is about 4%. Figure 11.40 shows the results of the preceding technique applied for transform coding of an image in the presence of channel errors. The improvement in SNR is 10 dB at $p = 0.01$ and is also significant visually. This scheme has been found to be quite robust with respect to fluctuations in the channel error rates [6, 39].

### 11.9 CODING OF TWO-TONE IMAGES

The need for electronic storage and transmission of graphics and two-tone images such as line drawings, letters, newsprint, maps, and other documents has been increasing rapidly, especially with the advent of personal computers and modern telecommunications. Commercial products for document transmission over telephone lines and data lines already exist. The CCITT† has recommended a set of eight documents (Fig. 11.41) for comparison and evaluation of different binary image coding algorithms. The CCITT standard sampling rates for typical A4 ($8 \frac{1}{2}$-in. by 11-in.) documents for transmission over the so-called Group 3 digital facsimile apparatus are 3.85 lines per millimeter at normal resolution and 7.7 lines per millimeter at high resolution in the vertical direction. The horizontal sampling rate standard is 1728 pixels per line, which corresponds to 7.7 lines per millimeter resolution or 200 points per inch (ppi). For newspaper pages and other documents that contain text as well as halftone images, sampling rates of 400 to 1000 ppi are used. Thus, for the standard $8 \frac{1}{2}$-in. by 11-in. page, $1.87 \times 10^6$ bits will be required at 200 ppi × 100 lpi sampling density. Transmitting this information over a 4800-bit/s telephone line will take over 6 min. Compression by a factor of, say, 5 can reduce the transmission time to about 1.3 minutes.

Many compression algorithms for binary images exploit the facts that (1) most pixels are white and (2) the black pixels occur with a regularity that manifests itself in the form of characters, symbols, or connected boundaries. There are three basic concepts of coding such images: (1) coding only transition points between black and white, (2) skipping white, and (3) pattern recognition. Figure 11.42 shows a convenient classification of algorithms based on these concepts.

#### Run-Length Coding

In run-length coding (RLC) the lengths of black and white runs on the scan lines are coded. Since white (1s) and black (0s) runs alternate, the color of the run need not

† Comité Consultatif International de Téléphonie et Télégraphie.
Dear Pete,

Thanks for your letter introducing me to the facility of facsimile transmission.

In facsimile a photograph is captured on a master scan and transmitted as a series of dots on a smaller copy produced at the remote destination. As a result, a facsimile copy of the master document is produced.

Probably you have used this facility in your organisation.

Yours sincerely,

Phil.

P.S. Croix
Group Leader - Terame Research

---

**Document 1**

**Document 2**

---

**Document 3**

**Document 4**

---

**Figure 11.41 CCITT test documents.**
Cela est d'autant plus visible que l'effet est plus visible. Ainsi, si la fréquence de l'oscillateur est modulée, l'amplitude des ondes sinusoidales est modulée de l'amplitude des ondes cosmiques pour représenter la figure 3.

Dans ce cas, la fréquence adaptée pour le cas échéant est confirmée par la figure 3, par la courbe

Une ligne droite est tracée dans l'aire d'une surface sinusoidale pour représenter la relation entre la fréquence et la phase modulée en la figure 3.

La ligne tracée est donnée par

\[ f = \frac{1}{T} \]

En cette phase est bien l'adaptation de l'aire du

à un décalage constant de la même amplitude et à un retard \( T \) pour produire

Un signal de même fréquence est analysé dans la zone d'un réseau \( F \) dans un décalage de l'oscillator de type, considéré entre \( F \) et \( F + \Delta F \), et celui du point d'intersection de \( F \) et \( F + \Delta F \) est donné en phase avec le signal de l'oscillator cisaillé. L'aire de la surface sinusoidale est représentée sur la figure 3. En ce cas en la figure 3, les points correspondant à la partie du réseau parabolique. On comprend le bien du comportement des ondes sinusoidales et ce signal de la fréquence adaptée est loisible à \( T \) et la phase adaptée dans l'équation. L'aire du réseau parabolique est représentée sur la figure 3.

On peut physiquement le phénomène de ce processus en utilisant une méthode qui est la figure 3. L'aire du réseau parabolique est tracée dans l'aire d'une surface sinusoidale pour représenter la relation entre la fréquence de l'oscillator et la phase modulée en la figure 3.
be coded (Fig. 11.43). The first run is always a white run with length zero, if necessary. The run lengths can be coded by fixed-length \( m \)-bit code words, each representing a block of maximum run length \( M - 1 \), \( M \geq 2^m \), where \( M \) can be optimized to maximize compression (see Section 11.2 and Problem 11.17).

A more-efficient technique is to use Huffman coding. To avoid a large code book, the truncated and the modified Huffman codes are used. The truncated Huffman code assigns separate Huffman code words for white and black runs up to lengths \( L_w \) and \( L_b \), respectively. Typical values of these runs have been found to be \( L_w = 47 \), \( L_b = 15 \) [1c, p. 1426]. Longer runs, which have lower probabilities, are assigned by a fixed-length code word, which consists of a prefix code plus an 11-bit binary code of the run length.

The modified Huffman code, which has been recommended by the CCITT as a one-dimensional standard code for Group 3 facsimile transmission, uses \( L_w = L_b = 63 \). Run lengths smaller than 64 are Huffman coded to give the terminator code. The remaining runs are assigned two code words, consisting of a make-up code and a terminator code. Table 11.10 gives the codes for the Group 3 standard. It also gives the end-of-line (EOL) code and an extended code table for larger paper widths up to A3 in size, which require up to 2560 pixels per line.

Other forms of variable-length coding that simplify the coding-decoding procedures are algorithm based. Noteworthy among these codes are the \( A_N \) and \( B_N \) codes [1c, p. 1406]. The \( A_N \) codes, also called \( L_N \) codes, are multiple fixed-length codes that are nearly optimal for exponentially distributed run lengths. They belong to a class of linear codes whose length increases approximately linearly with the number of messages. If \( l_k \), \( k = 1, 2, 3, \ldots \), are the run lengths, then the \( A_N \) code of block size \( N \) is obtained by writing \( k = q (2^N - 1) + r \), where \( 1 \leq r \leq 2^N - 1 \) and \( q \) is a
## Table 11.10 Modified Huffman Code Tables for One-dimensional Run-length Coding

<table>
<thead>
<tr>
<th>Run length</th>
<th>White runs</th>
<th>Black runs</th>
<th>Run length</th>
<th>White runs</th>
<th>Black runs</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00110101</td>
<td>0000110111</td>
<td>32</td>
<td>00011011</td>
<td>000001101010</td>
</tr>
<tr>
<td>1</td>
<td>000111</td>
<td>0100</td>
<td>33</td>
<td>00010110</td>
<td>000001101011</td>
</tr>
<tr>
<td>2</td>
<td>0111</td>
<td>110</td>
<td>34</td>
<td>00010111</td>
<td>000001101011</td>
</tr>
<tr>
<td>3</td>
<td>1000</td>
<td>010</td>
<td>35</td>
<td>00010100</td>
<td>000011010011</td>
</tr>
<tr>
<td>4</td>
<td>1011</td>
<td>011</td>
<td>36</td>
<td>00010101</td>
<td>000011010010</td>
</tr>
<tr>
<td>5</td>
<td>1100</td>
<td>001</td>
<td>37</td>
<td>00010110</td>
<td>000011010101</td>
</tr>
<tr>
<td>6</td>
<td>1110</td>
<td>0010</td>
<td>38</td>
<td>00010111</td>
<td>000011010110</td>
</tr>
<tr>
<td>7</td>
<td>1111</td>
<td>00011</td>
<td>39</td>
<td>00010010</td>
<td>000011011000</td>
</tr>
<tr>
<td>8</td>
<td>10011</td>
<td>000101</td>
<td>40</td>
<td>00010101</td>
<td>000011011001</td>
</tr>
<tr>
<td>9</td>
<td>10100</td>
<td>000100</td>
<td>41</td>
<td>00010110</td>
<td>000011011011</td>
</tr>
<tr>
<td>10</td>
<td>00111</td>
<td>000100</td>
<td>42</td>
<td>00010110</td>
<td>000110101101</td>
</tr>
<tr>
<td>11</td>
<td>01000</td>
<td>000101</td>
<td>43</td>
<td>00010110</td>
<td>000110111011</td>
</tr>
<tr>
<td>12</td>
<td>000011</td>
<td>000101</td>
<td>44</td>
<td>00010110</td>
<td>000110111011</td>
</tr>
<tr>
<td>13</td>
<td>000111</td>
<td>0000100</td>
<td>45</td>
<td>00010110</td>
<td>000110111011</td>
</tr>
<tr>
<td>14</td>
<td>110100</td>
<td>0000111</td>
<td>46</td>
<td>00010110</td>
<td>000110111011</td>
</tr>
<tr>
<td>15</td>
<td>110101</td>
<td>00001100</td>
<td>47</td>
<td>00010110</td>
<td>000110111011</td>
</tr>
<tr>
<td>16</td>
<td>101101</td>
<td>00001111</td>
<td>48</td>
<td>00010110</td>
<td>000110111011</td>
</tr>
<tr>
<td>17</td>
<td>101111</td>
<td>00011000</td>
<td>49</td>
<td>00010110</td>
<td>000110111011</td>
</tr>
<tr>
<td>18</td>
<td>0101111</td>
<td>00001100</td>
<td>50</td>
<td>00101100</td>
<td>000110111011</td>
</tr>
<tr>
<td>19</td>
<td>0011000</td>
<td>0000110111</td>
<td>51</td>
<td>00101100</td>
<td>000110111011</td>
</tr>
<tr>
<td>20</td>
<td>0001101</td>
<td>000111000</td>
<td>52</td>
<td>00101100</td>
<td>000110111011</td>
</tr>
<tr>
<td>21</td>
<td>0010111</td>
<td>0001110100</td>
<td>53</td>
<td>00101100</td>
<td>000110111011</td>
</tr>
<tr>
<td>22</td>
<td>0000111</td>
<td>0001111111</td>
<td>54</td>
<td>00101100</td>
<td>000110111011</td>
</tr>
<tr>
<td>23</td>
<td>0001010</td>
<td>0001110100</td>
<td>55</td>
<td>00101100</td>
<td>000110111011</td>
</tr>
<tr>
<td>24</td>
<td>0101000</td>
<td>0001110111</td>
<td>56</td>
<td>00101100</td>
<td>000110111011</td>
</tr>
<tr>
<td>25</td>
<td>0101011</td>
<td>0001111100</td>
<td>57</td>
<td>00101100</td>
<td>000110111011</td>
</tr>
<tr>
<td>26</td>
<td>0010101</td>
<td>00011101010</td>
<td>58</td>
<td>00101100</td>
<td>000110111011</td>
</tr>
<tr>
<td>27</td>
<td>0100100</td>
<td>00011110111</td>
<td>59</td>
<td>00101100</td>
<td>000110111011</td>
</tr>
<tr>
<td>28</td>
<td>0011000</td>
<td>00011110100</td>
<td>60</td>
<td>00101100</td>
<td>000110111011</td>
</tr>
<tr>
<td>29</td>
<td>00000010</td>
<td>00001110111</td>
<td>61</td>
<td>00110010</td>
<td>000110111011</td>
</tr>
<tr>
<td>30</td>
<td>00000101</td>
<td>0000111111</td>
<td>62</td>
<td>00110101</td>
<td>000110111011</td>
</tr>
<tr>
<td>31</td>
<td>00011010</td>
<td>00001101000</td>
<td>63</td>
<td>00110100</td>
<td>000110111011</td>
</tr>
</tbody>
</table>

A nonnegative integer. The codeword for $l_k$ has $(q + 1)N$ bits, of which the first $qN$ bits are 0 and the last $N$ bits are the binary representation of $r$. For example if $N = 2$, the $A_2$ code for $l_8$ ($8 = 2 \times 3 + 2$) is 000010. For a geometric distribution with mean $\mu$, the optimum $N$ is the integer nearest to $(1 + \log_2 \mu)$.

Experimental evidence shows that long run lengths are more common than predicted by an exponential distribution. A better model for run-length distribution is of the form

$$P(l) = \frac{c}{l^\alpha}, \quad \alpha > 0, c = \text{constant} \tag{11.120}$$
TABLE 11.10 (Continued)

<table>
<thead>
<tr>
<th>Run length</th>
<th>Make-up code words</th>
</tr>
</thead>
<tbody>
<tr>
<td>White runs</td>
<td>Black runs</td>
</tr>
<tr>
<td>64</td>
<td>000000111111</td>
</tr>
<tr>
<td>128</td>
<td>000011001000</td>
</tr>
<tr>
<td>192</td>
<td>000011001011</td>
</tr>
<tr>
<td>256</td>
<td>000011001011</td>
</tr>
<tr>
<td>320</td>
<td>000011001011</td>
</tr>
<tr>
<td>384</td>
<td>000011001011</td>
</tr>
<tr>
<td>448</td>
<td>000011001011</td>
</tr>
<tr>
<td>512</td>
<td>000011001011</td>
</tr>
<tr>
<td>576</td>
<td>000011001011</td>
</tr>
<tr>
<td>640</td>
<td>000011001011</td>
</tr>
<tr>
<td>704</td>
<td>000011001011</td>
</tr>
<tr>
<td>768</td>
<td>000011001011</td>
</tr>
<tr>
<td>832</td>
<td>000011001011</td>
</tr>
<tr>
<td>896</td>
<td>000011001011</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Black runs</th>
<th>White runs</th>
<th>Black runs</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>000001110000</td>
<td>0000000011000</td>
</tr>
<tr>
<td>128</td>
<td>000011001011</td>
<td>0000000011000</td>
</tr>
<tr>
<td>192</td>
<td>000011001011</td>
<td>0000000011000</td>
</tr>
<tr>
<td>256</td>
<td>000011001011</td>
<td>0000000011000</td>
</tr>
<tr>
<td>320</td>
<td>000011001011</td>
<td>0000000011000</td>
</tr>
<tr>
<td>384</td>
<td>000011001011</td>
<td>0000000011000</td>
</tr>
<tr>
<td>448</td>
<td>000011001011</td>
<td>0000000011000</td>
</tr>
<tr>
<td>512</td>
<td>000011001011</td>
<td>0000000011000</td>
</tr>
<tr>
<td>576</td>
<td>000011001011</td>
<td>0000000011000</td>
</tr>
<tr>
<td>640</td>
<td>000011001011</td>
<td>0000000011000</td>
</tr>
<tr>
<td>704</td>
<td>000011001011</td>
<td>0000000011000</td>
</tr>
<tr>
<td>768</td>
<td>000011001011</td>
<td>0000000011000</td>
</tr>
<tr>
<td>832</td>
<td>000011001011</td>
<td>0000000011000</td>
</tr>
<tr>
<td>896</td>
<td>000011001011</td>
<td>0000000011000</td>
</tr>
</tbody>
</table>

Extended Modified Huffman code table

<table>
<thead>
<tr>
<th>Run length (Black or White)</th>
<th>Make-up code word</th>
</tr>
</thead>
<tbody>
<tr>
<td>1792</td>
<td>0000000001000</td>
</tr>
<tr>
<td>1856</td>
<td>0000000001000</td>
</tr>
<tr>
<td>1920</td>
<td>0000000001101</td>
</tr>
<tr>
<td>1984</td>
<td>0000000001001000</td>
</tr>
<tr>
<td>2048</td>
<td>00000000010101000</td>
</tr>
<tr>
<td>2112</td>
<td>00000000010101000</td>
</tr>
<tr>
<td>2176</td>
<td>00000000010101000</td>
</tr>
<tr>
<td>2240</td>
<td>00000000010101000</td>
</tr>
<tr>
<td>2304</td>
<td>00000000010101000</td>
</tr>
<tr>
<td>2368</td>
<td>00000000010101000</td>
</tr>
<tr>
<td>2432</td>
<td>00000000010101000</td>
</tr>
<tr>
<td>2496</td>
<td>00000000010101000</td>
</tr>
<tr>
<td>2560</td>
<td>00000000010101000</td>
</tr>
</tbody>
</table>

which decreases less rapidly with \( l \) than the exponential distribution.

The \( B_N \) codes, also called \( H_N \) codes, are also multiples of fixed-length codes. Word length of \( B_N \) increases roughly as the logarithm of \( N \). The fixed block length for a \( B_N \) code is \( N + 1 \) bits. It is constructed by listing all possible \( N \)-bit words, followed by all possible \( 2N \)-bit words, and then \( 3N \)-bit words, and so on. An additional bit is inserted after every block of \( N \) bits. The inserted bit is 0 except for the bit inserted after the last block, which is 1.

Table 11.11 shows the construction of the \( B_1 \) code, which has been found useful for RLC.
### Example 11.10

Table 11.12 lists the averages $\mu_w$ and $\mu_b$ and the entropies $H_w$ and $H_b$ of white and black run lengths for CCITT documents. From this data an upper bound on achievable compression can be obtained as

$$C_{\text{max}} = \frac{\mu_w + \mu_b}{H_w + H_b}$$

which is also listed in the table. These results show compression factors of 5 to 20 are achievable by RLC techniques.

### White Block Skipping [1c, p. 1406, 44]

White block skipping (WBS) is a very simple but effective compression algorithm. Each scan line is divided into blocks of $N$ pixels. If the block contains all white pixels, it is coded by a 0. Otherwise, the code word has $N + 1$ bits, the first bit being 1, followed by the binary pattern of the block (Fig. 11.43). The bit rate for this method is

$$R_N = \frac{(1 - p_N)(N + 1) + p_N}{N}$$

$$= \left(1 - p_N + \frac{1}{N}\right) \text{ bits/pixel}$$

### Table 11.11 Construction of $B_1$ Code.

Inserted bits are underscored.

<table>
<thead>
<tr>
<th>Run length</th>
<th>$kN$-bit words</th>
<th>$B_1$ Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>$0_1$</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>$1_1$</td>
</tr>
<tr>
<td>3</td>
<td>0 0</td>
<td>$000_1$</td>
</tr>
<tr>
<td>4</td>
<td>0 1</td>
<td>$001_1$</td>
</tr>
<tr>
<td>5</td>
<td>1 0</td>
<td>$100_1$</td>
</tr>
<tr>
<td>6</td>
<td>1 1</td>
<td>$101_1$</td>
</tr>
<tr>
<td>7</td>
<td>000</td>
<td>$00000_1$</td>
</tr>
</tbody>
</table>

### Table 11.12 Run-Length Measurements for CCITT Documents

<table>
<thead>
<tr>
<th>Document number</th>
<th>$\mu_w$</th>
<th>$\mu_b$</th>
<th>$H_w$</th>
<th>$H_b$</th>
<th>$C_{\text{max}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>156.3</td>
<td>6.8</td>
<td>5.5</td>
<td>3.6</td>
<td>18.0</td>
</tr>
<tr>
<td>2</td>
<td>257.1</td>
<td>14.3</td>
<td>8.2</td>
<td>4.5</td>
<td>21.4</td>
</tr>
<tr>
<td>3</td>
<td>89.8</td>
<td>8.5</td>
<td>5.7</td>
<td>3.6</td>
<td>10.6</td>
</tr>
<tr>
<td>4</td>
<td>39.0</td>
<td>5.7</td>
<td>4.7</td>
<td>3.1</td>
<td>5.7</td>
</tr>
<tr>
<td>5</td>
<td>79.2</td>
<td>7.0</td>
<td>5.7</td>
<td>3.3</td>
<td>9.5</td>
</tr>
<tr>
<td>6</td>
<td>138.5</td>
<td>8.0</td>
<td>6.2</td>
<td>3.6</td>
<td>14.9</td>
</tr>
<tr>
<td>7</td>
<td>45.3</td>
<td>4.4</td>
<td>5.9</td>
<td>3.1</td>
<td>5.6</td>
</tr>
<tr>
<td>8</td>
<td>85.7</td>
<td>70.9</td>
<td>6.9</td>
<td>5.8</td>
<td>12.4</td>
</tr>
</tbody>
</table>
where $p_N$ is the probability that a block contains all white pixels. This rate depends on the block size $N$. The value $N = 10$ has been found to be suitable for a large range of images. Note that WBS is a simple form of the truncated Huffman code and should work well especially when an image contains large white areas.

An adaptive WBS scheme improves the performance significantly by coding all white scan lines separately. A 0 is assigned to an all-white scan line. If a line contains at least one black pixel, a 1 precedes the regular WBS code for that line. The WBS method can also be extended to two dimensions by considering $M \times N$ blocks of pixels. An all-white block is coded by a 0. Other blocks are coded by $(MN + 1)$ bits, whose first bit is 1 followed by the block bit pattern. An adaptive WBS scheme that uses variable block size proceeds as follows. If the initial block contains all white pixels, it is represented by 0. Otherwise, a prefix of 1 is assigned and the block is subdivided into several subblocks, each of which is then treated similarly. The process continues until an elementary block is reached, which is coded by the regular WBS method. Note that this method is very similar to generating the quad-tree code for a region (see Section 9.7).

**Prediction Differential Quantization** [lc, p. 1418]

Prediction differential quantization (PDQ) is an extension of run-length coding, where the correlation between scan lines is exploited. The method basically encodes the overlap information of a black run in successive scan lines. This is done by coding differences $\Delta'$ and $\Delta''$ in black runs from line to line together with messages, new start (NS) when black runs start, and merge (M) when there is no further overlap of that run. A new start is coded by a special code word, whereas a merge is represented by coding white and black runlengths $r_w, r_b$, as shown in Fig. 11.44.

**Relative Address Coding** [lf, p. 834]

Relative address coding (RAC) uses the same principle as the PDQ method and computes run-length differences by tracking either the last transition on the same line or the nearest transition on the previous line. For example, the transition pixel $Q$ (Fig. 11.45) is encoded by the shortest distance $PQ$ or $Q'Q$, where $P$ is the preceding transition element on the current line and $Q'$ is the nearest transition element to the right of $P$ on the previous line, whose direction of transition is the same as that of $Q$. If $P$ does not exist, then it is considered to be the imaginary pixel to the right of the last pixel on the preceding line. The distance $QQ'$ is coded as $+N$.

![Figure 11.44 The PDQ method: $\Delta'' = r_2 - r_1$.](image-url)

![Figure 11.45 RAC method.](image-url)
TABLE 11.13  Relative Address Codes. \( x \ldots x \) = binary representation of \( N \).

<table>
<thead>
<tr>
<th>Distance</th>
<th>Code</th>
<th>( N )</th>
<th>( F(N) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>+0</td>
<td>0</td>
<td>1-4</td>
<td>0xx</td>
</tr>
<tr>
<td>+1</td>
<td>100</td>
<td>5-20</td>
<td>10xxxx</td>
</tr>
<tr>
<td>-1</td>
<td>101</td>
<td>21-84</td>
<td>110xxxxxx</td>
</tr>
<tr>
<td>( N (N &gt; 1) )</td>
<td>111 ( F(N) )</td>
<td>85-340</td>
<td>1110xxxxxxx</td>
</tr>
<tr>
<td>(+N (N &gt; 2) )</td>
<td>1100 ( F(N) )</td>
<td>341-1364</td>
<td>11110xxxxxxx</td>
</tr>
<tr>
<td>(-N (N &gt; 2) )</td>
<td>1101 ( F(N) )</td>
<td>1365-5460</td>
<td>111110xxxxxxx</td>
</tr>
</tbody>
</table>

if \( Q' \) is \( N (\geq 0) \) pixels to the left or \(-N \) if \( Q' \) is \( N(\geq 1) \) pixels to the right of \( Q \) on the preceding line. Distance \( PQ \) is coded as \( N(\geq 1) \) if it is \( N \) pixels away. The RAC distances are coded by a code similar to the \( B_1 \) code, except for the choice of the reference line and for very short distances, +0, +1, −1 (see Table 11.13).

**CCITT Modified Relative Element Address Designate Coding**

The modified relative element address designate (READ) algorithm has been recommended by CCITT for two-dimensional coding of documents. It is a modification of the RAC and other similar codes [1f, p. 854]. Referring to Fig. 11.46 we define \( a_0 \) as the reference transition element whose position is defined by the previous coding mode (to be discussed shortly). Initially, \( a_0 \) is taken to be the imaginary white transition pixel situated to the left of the first pixel on the coding

![Figure 11.46 CCITT modified READ coding.](image-url)
line. The next pair of transition pixels to the right of \( a_0 \) are labeled \( a_1 \) and \( a_2 \) on the coding line and \( b_1 \) and \( b_2 \) on the reference line and have alternating colors to \( a_0 \). Any of the elements \( a_1, a_2, b_1, b_2 \) not detected for a particular coding line is taken as the imaginary pixel to the right of the last element of its respective scan line. Pixel \( a_1 \) represents the next transition element to be coded. The algorithm has three modes of coding, as follows.

**Pass mode.** \( b_2 \) is to the left of \( a_1 \) (Fig. 11.46a). This identifies the white or black runs on the reference line that do not overlap with the corresponding white or black runs on the coding line. The reference element \( a_0 \) is set below \( b_2 \) in preparation for the next coding.

**Vertical mode.** \( a_1 \) is coded relative to \( b_1 \) by the distance \( a_1 b_1 \), which is allowed to take the values 0, 1, 2, 3 to the right or left of \( b_1 \). These are represented by \( V(0), V_R(x), V_L(x), x = 1, 2, 3 \) (Table 11.14). In this mode \( a_0 \) is set at \( a_1 \) in preparation for the next coding.

**Horizontal mode.** If \( |a_1 b_1| > 3 \), the vertical mode is not used and the run lengths \( a_0 a_1 \) and \( a_1 a_2 \) are coded using the modified Huffman codes of Table 11.10. After coding, the new position of \( a_0 \) is set at \( a_2 \). If this mode is needed for the first element on the coding line, then the value \( a_0 a_1 = 1 \) rather than \( a_0 a_1 \) is coded. Thus if the first element is black, then a run length of zero is coded.

### Table 11.14 CCITT Modified READ Code Table [1f, p. 865]

<table>
<thead>
<tr>
<th>Mode</th>
<th>Elements to be coded</th>
<th>Notation</th>
<th>Code Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pass</td>
<td>( b_1, b_2 )</td>
<td>( P )</td>
<td>0001</td>
</tr>
<tr>
<td>Horizontal</td>
<td>( a_0 a_1, a_1 a_2 )</td>
<td>( H )</td>
<td>001 + ( M(a_0 a_1) + M(a_1 a_2) )</td>
</tr>
<tr>
<td>Vertical</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>( a_1 ) just under ( b_1 )</td>
<td>( a_1 b_1 = 0 )</td>
<td>( V(0) ) 1</td>
</tr>
<tr>
<td></td>
<td>( a_1 ) to the right of ( b_1 )</td>
<td>( a_1 b_1 = 1 )</td>
<td>( V_R(1) ) 011</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( a_1 b_1 = 2 )</td>
<td>( V_R(2) ) 000011</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( a_1 b_1 = 3 )</td>
<td>( V_R(3) ) 0000011</td>
</tr>
<tr>
<td></td>
<td>( a_1 ) to the left of ( b_1 )</td>
<td>( a_1 b_1 = 1 )</td>
<td>( V_L(1) ) 010</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( a_1 b_1 = 2 )</td>
<td>( V_L(2) ) 000010</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( a_1 b_1 = 3 )</td>
<td>( V_L(3) ) 0000010</td>
</tr>
</tbody>
</table>

2-D extensions
1-D extensions
End-of-line (EOL) code word
1-D coding of next line
2-D coding of next line

\( M(a_0 a_1) \) and \( M(a_1 a_2) \) are code words taken from the modified Huffman code tables given in Table 11.10. The bit assignment for the \( xxx \) bits is 111 for the uncompressed mode.
Figure 11.47  CCITT modified READ coding algorithm.
The coding procedure along a line continues until the imaginary transition element to the right of the last actual element on the line has been detected. In this way exactly 1728 pixels are coded on each line. Figure 11.47 shows the flow diagram for the algorithm. Here $K$ is called the $K$-factor, which means that after a one-dimensionally coded line, no more than $K - 1$ successive lines are two-dimensionally coded. CCITT recommended values for $K$ are 2 and 4 for documents scanned at normal resolution and high resolution, respectively. The $K$-factor is used to minimize the effect of channel noise on decoded images. The one-dimensional and two-dimensional extension code words listed in Table 11.14, with $xxx$ equal to 111, are used to allow the coder to enter the uncompressed mode, which may be desired when the run lengths are very small or random, such as in areas of halftone images or cross hatchings present in some business forms.

Predictive Coding

The principles of predictive coding can be easily applied to binary images. The main difference is that the prediction error is also a binary variable, so that a quantizer is not needed. If the original data has redundancy, then the prediction error sequence will have large runs of 0s (or 1s). For a binary image $u(m, n)$, let $\bar{u}(m, n)$ denote its predicted value based on the values of pixels in a prediction window $W$, which contains some of the previously coded pixels. The prediction error is defined as

$$
e(m, n) = \begin{cases} 1, & \bar{u}(m, n) \neq u(m, n) \\ 0, & \bar{u}(m, n) = u(m, n) \end{cases} = u(m, n) \oplus \bar{u}(m, n) \quad (11.123)$$

The sequence $e(m, n)$ can be coded by a run-length or entropy coding method. The image is reconstructed from $e(m, n)$ simply as

$$u(m, n) = \bar{u}(m, n) \oplus e(m, n) \quad (11.124)$$

Note that this is an errorless predictive coding method. An example of a prediction window $W$ for a raster scanned image is shown in Fig. 11.48.

A reasonable prediction criterion is to minimize the prediction error probability. For an $N$-element prediction window, there are $2^N$ different states. Let $S_k$, $k = 1, 2, \ldots, 2^N$ denote the $k$th state of $W$ with probability $p_k$ and define

$$q_k = \text{Prob}[u(m, n) = 1|S_k] \quad (11.125)$$

Then the optimum prediction rule having minimum prediction error probability is

$$\bar{u}(m, n) = \begin{cases} 1, & \text{if } q_k \geq 0.5 \\ 0, & \text{if } q_k < 0.5 \end{cases} \quad (11.126)$$

If the random sequence $u(m, n)$ is strict-sense stationary, then the various probabilities will remain constant at every $(m, n)$, and therefore the prediction rule stays the same. In practice a suitable choice of $N$ has to be made to achieve a trade-off between prediction error probability and the complexity of the predictor.
Figure 11.48 TUH method of predictive coding. The run length \( l_k \) of state \( S_k \) means a prediction error has occurred after state \( S_k \) has repeated \( l_k \) times.

due to large values of \( N \). Experimentally, 4 to 7 pixel predictors have been found to be adequate. Corresponding to the prediction rule of (11.126), the minimized prediction error is

\[
p_e = \sum_{k=1}^{2^N} p_k \min(q_k, 1 - q_k)
\]

(11.127)

If the random sequence \( u(m, n) \) is Markovian with respect to the prediction window \( W \), then the run lengths for each state \( S_k \) are independent. Hence, the prediction-error run lengths for each state (Fig. 11.47) can be coded by the truncated Huffman code, for example. This method has been called the Technical University of Hannover (TUH) code [1c, p. 1425].

**Adaptive Predictors**

Adaptive predictors are useful in practice because the image data is generally nonstationary. In general, any pattern classifier or a discriminant function could be used as a predictor. A simple classifier is a linear learning machine or adaptive threshold logic unit (TLU), which calculates the threshold \( q_k \) as a linear functional of the states of the pixels in the prediction window. Another type of pattern classifier is a network of TLUs called layered machines and includes piecewise linear discriminant functions and the so-called \( \alpha \)-perceptron. A practical adaptive predictor uses a counter \( C_k \) of \( L \) bits for each state [43]. The counter runs from 0 to \( 2^L - 1 \). The adaptive prediction rule is

\[
\bar{u}(m, n) = \begin{cases} 
1, & \text{if } C_k \geq 2^L - 1 \\
0, & \text{if } C_k < 2^L - 1 
\end{cases}
\]

(11.128)

After prediction of a pixel has been performed, the counter is updated as

\[
C_k = \begin{cases} 
\min(C_k + 1, 2^L - 1), & \text{if } u(m, n) = 1 \\
\max(C_k - 1, 0), & \text{otherwise}
\end{cases}
\]

(11.129)
### TABLE 11.15  Compression Ratios of Different Binary Coding Algorithms

<table>
<thead>
<tr>
<th>Document</th>
<th>One-dimensional codes</th>
<th>Two-dimensional codes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$B_1$ code</td>
<td>Truncated Huffman code</td>
</tr>
<tr>
<td>1</td>
<td>13.62</td>
<td>17.28</td>
</tr>
<tr>
<td>2</td>
<td>14.45</td>
<td>15.05</td>
</tr>
<tr>
<td>3</td>
<td>8.00</td>
<td>8.88</td>
</tr>
<tr>
<td>4</td>
<td>4.81</td>
<td>5.74</td>
</tr>
<tr>
<td>5</td>
<td>7.67</td>
<td>8.63</td>
</tr>
<tr>
<td>6</td>
<td>9.78</td>
<td>10.14</td>
</tr>
<tr>
<td>7</td>
<td>4.60</td>
<td>4.69</td>
</tr>
<tr>
<td>8</td>
<td>8.54</td>
<td>7.28</td>
</tr>
<tr>
<td>Average</td>
<td>8.93</td>
<td>9.71</td>
</tr>
</tbody>
</table>

The value $L = 3$ has been found to yield minimum prediction error for a typical printed page.

**Comparison of Algorithms**

Table 11.15 shows a comparison of the compression ratios achievable by different algorithms. The compression ratios for one-dimensional codes are independent of the vertical resolution. At normal resolution the two-dimensional codes improve the compression by only 10 to 30% over the modified Huffman code. At high resolution the improvements are 40 to 60% and are significant enough to warrant the use of these algorithms. Among the two-dimensional codes, the TUH predictive code is superior to the relative address techniques, especially for text information. However, the latter are simpler to code. The CCITT READ code, which is a modification of the RAC, performs somewhat better.

**Other Methods**

Algorithms that utilize higher-level information, such as whether the image contains a known type (or font) of characters or graphics, line drawings, and the like, can be designed to obtain very high-compression ratios. For example, in the case of printed text limited to the 128 ASCII characters, for instance, each character can be coded by 7 bits. The coding technique would require a character recognition algorithm. Likewise, line drawings can be efficiently coded by boundary-following algorithms, such as chain codes, line segments, or splines. Algorithms discussed here are not directly useful for halftone images because the image area has been modulated by pseudorandom noise and thresholded thereafter. In all these cases special preprocessing and segmentation is required to code the data efficiently.

**11.10 COLOR AND MULTISPECTRAL IMAGE CODING**

Data compression techniques discussed so far can be generalized to color and multispectral images, as shown in Fig. 11.49. Each pixel is represented by a $p \times 1$
vector. For example, in the case of color, the input is a $3 \times 1$ vector containing the $R, G, B$ components. This vector is transformed to another coordinate system, where each component can be processed by an independent spatial coder.

In coding color images, consideration should be given to the facts that (1) the luminance component ($Y$) has higher bandwidth than the chrominance components ($I, Q$) or ($U, V$) and (2) the color-difference metric is non-Euclidean in these coordinates, that is, equal noise power in different color components is perceived differently. In practical image coding schemes, the lower-bandwidth chrominance signals are sampled at correspondingly lower rates. Typically, the $I$ and $Q$ signals are sampled at one-third and one-sixth of the sampling rate of the luminance signal. Use of color-distance metric(s) is possible but has not been used in practical systems primarily because of the complexity of the color vision model (see Chapter 3).

An alternate method of coding color images is by processing the composite color signal. This is useful in broadcast applications, where it is desired to manage only one signal. However, since the luminance and color signals are not in the same frequency band, the foregoing monochrome image coding techniques are not very efficient if applied directly. Typically, the composite signal is sampled at $3f_{sc}$ (the lowest integer multiple of subcarrier frequency above the Nyquist rate) or $4f_{sc}$, and

![Figure 11.50 Subcarrier phase relationships in sampled NTSC signal. Pixels having the same labels have the same subcarrier phase.](image-url)
the designs of predictors (for DPCM coding) or the block size (for transform coding) take into account the relative phases of the pixels. For example, at $3f_{sc}$ sampling, the adjacent samples of subcarrier have $120^\circ$ phase difference and at $4f_{sc}$ sampling, the phase difference is $90^\circ$. Figure 11.50 shows the subcarrier phase relationships of neighboring pixels in the sampled NTSC signal. Due to the presence of the modulated subcarrier, higher-order predictors are required for DPCM of composite signals. Table 11.16 gives examples of predictors for predictive coding of the NTSC signal.

Table 11.17 lists the practical bit rates achievable by different coding algorithms for broadcast quality reproduced images. These results show that the chrominance components can be coded by as few as $\frac{1}{2}$ bit per pixel (via adaptive transform coding) to 1 bit per pixel (via DPCM).

Due to the flexibility in the design of coders, component coding performs somewhat better than composite coding and may well become the preferred choice with the advent of digital television.

For multispectral images, the input data is generally KL transformed in the temporal direction to obtain the principal components (see Section 7.6). Each

---

**Table 11.16** Predictors for DPCM of Composite NTSC Signal. $z^{-1} = 1$ pixel delay, $z^{-N} = 1$ line delay, $z^{-262N} = 1$ field delay, $p$ (leak) $\leq 1$.

<table>
<thead>
<tr>
<th>Sampling rate</th>
<th>Predictor, $P(z)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2f_{sc}$</td>
<td>1-D</td>
</tr>
<tr>
<td></td>
<td>2-D</td>
</tr>
<tr>
<td>$3f_{sc}$</td>
<td>1-D</td>
</tr>
<tr>
<td></td>
<td>2-D</td>
</tr>
<tr>
<td></td>
<td>3-D</td>
</tr>
<tr>
<td>$4f_{sc}$</td>
<td>1-D</td>
</tr>
<tr>
<td></td>
<td>2-D</td>
</tr>
<tr>
<td></td>
<td>3-D</td>
</tr>
</tbody>
</table>

**Table 11.17** Typical Performance of Component Coding Algorithms on Color Images

<table>
<thead>
<tr>
<th>Method</th>
<th>Components coded</th>
<th>Description</th>
<th>Rate per component/pixel</th>
<th>Average rate bits/pixel</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCM</td>
<td>$R, G, B$</td>
<td>Raw data</td>
<td>8</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>$U^<em>, V^</em>, W^*$</td>
<td>Color space quantizer</td>
<td>1024 color cells</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>$Y, I, Q$</td>
<td>$I, Q$ subsampled</td>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>DPCM One-step predictor</td>
<td>$Y, I, Q$</td>
<td>$I, Q$ subsampled</td>
<td>2 to 3</td>
<td>3 to 4.5</td>
</tr>
<tr>
<td>Transform (cosine, slant)</td>
<td>$Y, I, Q$</td>
<td>No subsampling</td>
<td>$Y$ (1.75 to 2)</td>
<td>2.5 to 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$I, Q$ (0.75 to 1)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$Y, I, Q$</td>
<td>Same as above with</td>
<td>Variable</td>
<td>1 to 2</td>
</tr>
<tr>
<td></td>
<td>$Y, U, V$</td>
<td>adaptive classification</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Sec. 11.10 Color and Multispectral Image Coding 555
TABLE 11.18  Summary of Image Data Compression Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Typical average rates</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>bits/pixel</td>
<td></td>
</tr>
<tr>
<td><strong>Zero-memory methods</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCM</td>
<td>6-8</td>
<td>Simple to implement.</td>
</tr>
<tr>
<td>Contrast quantization</td>
<td>4-5</td>
<td></td>
</tr>
<tr>
<td>Pseudorandom noise—quantization</td>
<td>4-5</td>
<td></td>
</tr>
<tr>
<td>Line interlace</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Dot interlace</td>
<td>2-4</td>
<td></td>
</tr>
<tr>
<td><strong>Predictive coding</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delta modulation</td>
<td>1</td>
<td>Performance poorer than DPCM, oversample data for improvement.</td>
</tr>
<tr>
<td>Intraframe DPCM</td>
<td>2-3</td>
<td>Predictive methods are generally simple to implement, but sensitive to data statistics. Adaptive techniques improve performance substantially. Channel error effects are cumulative and visibly degrade image quality.</td>
</tr>
<tr>
<td>Intraframe adaptive DPCM</td>
<td>1-2</td>
<td></td>
</tr>
<tr>
<td>Interframe conditional—replenishment</td>
<td>1-2</td>
<td></td>
</tr>
<tr>
<td>Interframe DPCM</td>
<td>1-1.5</td>
<td></td>
</tr>
<tr>
<td>Interframe adaptive DPCM</td>
<td>0.5-1</td>
<td></td>
</tr>
<tr>
<td><strong>Transform coding</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intraframe</td>
<td>1-1.5</td>
<td>Achieve high performance, small sensitivity to fluctuation in data statistics, channel and quantization errors distributed over the image block. Easy to provide channel protection. Hardware complexity is high.</td>
</tr>
<tr>
<td>Intraframe adaptive</td>
<td>0.5-1</td>
<td></td>
</tr>
<tr>
<td>Interframe</td>
<td>0.5-1</td>
<td></td>
</tr>
<tr>
<td>Interframe adaptive</td>
<td>0.1-0.5</td>
<td></td>
</tr>
<tr>
<td><strong>Hybrid coding</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intraframe</td>
<td>1-2</td>
<td>Achieve performance close to transform coding at moderate rates (0.5 to 1 bit/pixel). Complexity lies midway between transform coding and DPCM.</td>
</tr>
<tr>
<td>Intraframe adaptive</td>
<td>0.5-1.5</td>
<td></td>
</tr>
<tr>
<td>Interframe</td>
<td>0.5-1</td>
<td></td>
</tr>
<tr>
<td>Interframe adaptive</td>
<td>0.25-0.5</td>
<td></td>
</tr>
<tr>
<td><strong>Color image coding</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intraframe</td>
<td>1-3</td>
<td>The above techniques are applicable.</td>
</tr>
<tr>
<td>Interframe</td>
<td>0.25-1</td>
<td></td>
</tr>
<tr>
<td><strong>Two-tone image coding</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-D Methods</td>
<td>0.06-0.2</td>
<td>Distortionless coding. Higher compression achievable by pattern recognition techniques.</td>
</tr>
<tr>
<td>2-D Methods</td>
<td>0.03-0.2</td>
<td></td>
</tr>
</tbody>
</table>
component is independently coded by a two-dimensional coder. An alternative is to identify a finite number of clusters in a suitable feature space. Each multispectral pixel is represented by the information pertaining to (e.g., the centroid of) the cluster to which it belongs. Usually the fidelity criterion is the classification (rather than mean square) accuracy of the encoded data [45, 46].

11.11 SUMMARY

Image data compression techniques are of significant practical interest. We have considered a large number of compression algorithms that are available for implementation. We conclude this chapter by providing a summary of these methods in Table 11.18.

PROBLEMS

11.1* For an 8-bit integer image of your choice, determine the Nth-order prediction error field \( e_N(m, n) = u(m, n) - \bar{u}_N(m, n) \) where \( \bar{u}_N(m, n) \) is the best mean square causal predictor based on the \( N \) nearest neighbors of \( u(m, n) \). Truncate \( \bar{u}_N(m, n) \) to the nearest integer and calculate the entropy of \( e_N(m, n) \) from its histogram for \( N = 0, 1, 2, 3, 4, 5 \). Using these as estimates for the Nth-order entropies, calculate the achievable compression.

11.2 The output of a binary source is to be coded in blocks of \( M \) samples. If the successive outputs are independent and identically distributed with \( p = 0.95 \) (for a 0), find the Huffman codes for \( M = 1, 2, 3, 4 \) and calculate their efficiencies.

11.3 For the AR sequence of (11.25), the predictor for feedforward predictive coding (Fig 11.6) is chosen as \( \bar{u}(n) = \rho u(n - 1) \). The prediction error sequence \( e(n) = u(n) - \bar{u}(n) \) is quantized using \( B \) bits. Show that in the steady state,

\[
E[|\delta u(n)|^2] = \frac{\sigma_u^2}{1 - \rho^2} = \sigma_e^2 f(B)
\]

where \( \sigma_e^2 = \sigma_u^2 (1 - \rho^2)f(B) \) is the mean square quantization error of \( e(n) \). Hence the feedforward predictive coder cannot perform better than DPCM because the preceding result shows its mean square error is precisely the same as in PCM. This result happens to be true for arbitrary stationary sequences utilizing arbitrary linear predictors. A possible instance where the feedforward predictive coder may be preferred over DPCM is in the distortionless case, where the quantizer is replaced by an entropy coder. The two coders will perform identically, but the feedforward predictive coder will have a somewhat simpler hardware implementation.

11.4 (Delta modulation analysis) For delta modulation of the AR sequence of (11.25), write the prediction error as \( e(n) = \epsilon(n) - (1 - \rho)u(n - 1) + \delta e(n - 1) \). Assuming a 1-bit Lloyd-Max quantizer and \( \delta e(n) \) to be an uncorrelated sequence, show that

\[
\sigma_e^2(n) = 2(1 - \rho)\sigma_u^2 + (2\rho - 1)\sigma_e^2(n - 1) f(1)
\]

from which (11.26) follows after finding the steady-state value of \( \sigma_u^2/\sigma_e^2(n) \).
11.5 Consider images with power spectral density function

\[ S(z_1, z_2) = \frac{\beta^2}{A(z_1, z_2)A(z_1^{-1}, z_2^{-1})} \]

\[ A(z_1, z_2) \triangleq 1 - P(z_1, z_2) \]

where \( A(z_1, z_2) \) is the minimum-variance, causal, prediction-error filter and \( \beta^2 \) is the variance of the prediction error. Show that the DPCM algorithm discussed in the text takes the form shown in Fig. P11.5, where \( H(z_1, z_2) = 1/A(z_1, z_2) \). If the channel adds independent white noise of variance \( \sigma_w^2 \), show that the total noise in the reconstructed output would be \( \sigma_w^2 = \sigma_q^2 + (\sigma_q^2 \sigma_e^2)/\beta^2 \), where \( \sigma_q^2 \triangleq E[(u(m, n))^2] \).

\[ H(z_1, z_2) \]

\[ \rightarrow \quad \text{Decoder} \]

\[ \hat{u}(m, n) \]

\[ u(m, n) \]

\[ + \]

\[ e \]

\[ e' \]

\[ \text{Quantizer} \]

\[ \hat{e} \]

\[ \text{Coder} \]

\[ P(z_1, z_2) \]

\[ \text{Channel noise} \]

\[ u(m, n) \]

Figure P11.5

11.6 (DPCM analysis) For DPCM of the AR sequence of (11.25), write the prediction error as \( e(n) = e(n) + \rho \delta e(n - 1) \). Assuming \( \delta e(n) \) to be an uncorrelated sequence, show that the steady-state distortion due to DPCM is

\[ D \triangleq E[\delta e^2(n)] = \sigma_q^2 (1 - \rho^2) f(B) \]

For \( \rho = 0.95 \), plot the normalized distortion \( D/\sigma_w^2 \) as a function of bit rate \( B \) for \( B = 1, 2, 3, 4 \) for a Laplacian density quantizer and compare it with PCM.

11.7* For a 512 \( \times \) 512 image of your choice, design DPCM coders using mean square predictors of orders up to four. Implement the coders for \( B = 3 \) and compare the reconstructed images visually as well as on the basis of their mean square errors and entropies.

11.8 a. Using the transform coefficient variances given in Table 5.2 and the Shannon quantizer based rate distortion formulas (11.61) to (11.63), compare the distortion versus rate curves for the various transforms. (Hint: An easy way is to arrange \( \sigma_j^2 \) in decreasing order and let \( \theta = \sigma_j^2, \quad j = 0, \ldots, N - 1 \) and plot \( D_j \triangleq 1/N \sum_{j=0}^{N-1} \sigma_j^2 \) versus \( R_j \triangleq 1/2N \sum_{k=0}^{N-1} \log_2 \sigma_j^2/\sigma_j^2 \).

b. Compare the cosine transform \( R \) versus \( D \) function when the bit allocation is determined first by truncating the real numbers obtained via (11.61) to nearest integers and second, using the integer bit allocation algorithm.

11.9 (Whitening transform versus unitary transform) An \( N \times 1 \) vector \( u \) with covariance \( R = \{p^{m-n}\} \) is transformed as \( v = Lu \), where \( L \) is a lower triangular (nonunitary) matrix whose elements are

\[ l_{i,j} = \begin{cases} 1, & i = j \\ -\rho, & i - j = 1 \\ 0, & \text{otherwise} \end{cases} \]
a. Show that \( v \) is a vector of uncorrelated elements with \( \sigma^2(0) = 1 \) and \( \sigma^2(k) = 1 - \rho^2, \ 1 \leq k \leq N - 1 \).

b. If \( v(k) \) is quantized using \( n_k \) bits, show the reproduced vector \( u \) has the mean square distortion

\[
D = \frac{1}{N} \sum_{k=0}^{N-1} w_k f(n_k), \quad w_0 \Delta = \frac{(1 - \rho^{2N})}{1 - \rho^2}, \quad w_k \Delta = 1 - \rho^{2(N-k)}
\]

c. For \( N = 15, \rho = 0.95, \) and \( f(x) = 2^{-2x} \), find the optimum rate versus distortion function of this coder and compare it with that of the cosine transform coder.

From these results conclude that it is more advantageous to replace the (usually slow) KL transform by a fast unitary transform rather than a fast decorrelating transform.

11.10 (Transform coding versus DPCM) Suppose a sequence of length \( N \) has a causal representation

\[
u(n) = \tilde{u}(n) + \epsilon(n) = \sum_{k=0}^{n-1} a(n, k)u(k) + \epsilon(n), \quad 0 \leq n \leq N - 1
\]

where \( \tilde{u}(n) \) is the optimum linear predictor of \( u(n) \) and \( \epsilon(n) \) is an uncorrelated sequence of variance \( \beta_0^2 \). Writing this in vector notation as \( Lu = e, \epsilon(0) \Delta u(0) \), where \( L \) is an \( N \times N \) unit lower triangular matrix, it can be shown that \( R \), the covariance matrix of \( u \), satisfies \( |R| = \prod_{k=0}^{N-1} \beta_k^2 \). If \( u(n) \) is DPCM coded, then for small levels of distortion, the average minimum achievable rate is

\[
R_{DPCM} = \frac{1}{2N} \sum_{k=0}^{N-1} \log_2 \frac{\sigma^2(k)}{D}, \quad D < \min_k \{\beta_k^2\}
\]

a. Show that at small distortion levels, the above results imply

\[
R_{DPCM} > \frac{1}{2N} \sum_{k=0}^{N-1} \log_2 \frac{\beta_k^2}{D} = \frac{1}{2N} \log_2 |R| - \frac{1}{2} \log_2 D = R_{KL}
\]

that is, the average minimum rate achievable by KL transform coding is lower than that of DPCM.

b. Suppose the Markov sequence of (11.25) with \( \sigma^2 = 1, \beta_0^2 = 1, \beta_n^2 = (1 - \rho^2) \) for \( 1 \leq n \leq N - 1 \), is DPCM coded and the steady state exists for \( n \geq 1 \). Using the results of Problem 11.6, with \( f(B) = 2^{-2B} \), show that the number of bits \( B \) required to achieve a distortion level \( D \) is given by

\[
B = \frac{1}{2} \log_2 \left( \rho^2 + \frac{(1 - \rho^2)}{D} \right), \quad 1 \leq n \leq N - 1.
\]

For the initial sample \( \epsilon(0) \), the same distortion level is achieved by using \( \frac{1}{2} \log_2 (1/D) \) bits. From these and (11.69) show that

\[
R_{DPCM} = \frac{1}{2N} \log_2 \frac{1}{D} + \frac{(N-1)}{2N} \log_2 \left( \rho^2 + \frac{(1 - \rho)}{D} \right)
\]

\[
R_{DPCM} - R_{KL} = \frac{(N-1)}{2N} \log_2 \left( 1 + \frac{\rho^2 D}{(1 - \rho^2)} \right), \quad D < \frac{(1 - \rho)}{1 + \rho}
\]

Calculate this difference for \( N = 16, \rho = 0.95, \) and \( D = 0.01 \), and conclude that at low levels of distortion the performance of KL transform and DPCM coders
is close for Markov sequences. This is a useful result, which can be generalized for AR sequences. For ARMA sequences, bandlimited sequences, and two-dimensional random fields, this difference can be more significant.

11.11 For the separable covariance model used in Example 11.5, with $\rho = 0.95$, plot and compare the $R$ versus $D$ performances of (a) various transform coders for $16 \times 16$ size block utilizing Shannon quantizers (Hint: Use the data of Table 5.2.) and (b) $N \times N$ block cosine transform coders with $N = 2^n$, $n = 1, 2, \ldots, 8$. (Hint: Use eq. (P5.28–2).]

11.12 Plot and compare the $R$ versus $D$ curves for $16 \times 16$ block transform coding of images modeled by the nonseparable exponential covariance function $0.95 \sqrt{m^2+n^2}$ using the discrete, Fourier, cosine, sine, Hadamard, slant, and Haar transforms. (Hint: Use results of Problem P5.29 to calculate transform domain variances.)

11.13* Implement the zonal transform coding algorithm of Section 11.5 on $16 \times 16$ blocks of an image of your choice. Compare your results for average rates of 0.5, 1.0, and 2.0 bits per pixel using the cosine transform or any other transform of your choice.

11.14* Develop a chart of adaptive transform coding algorithms containing details of the algorithms and their relative merits and complexities. Implement your favorite of these and compare it with the $16 \times 16$ block cosine transform coding algorithm.

11.15 The motivation for hybrid coding comes from the following example. Suppose an $N \times N$ image $u(m, n)$ has the autocorrelation function $r(k, l) = \rho^{|k|+|l|}$.

a. If each column of the image transformed as $v_n = \Phi u_n$, where $\Phi$ is the KLT of $u_n$, then show that the autocorrelation of $v_n$, that is, $E[v_n(k)v_n(k')] = \lambda_k \rho^{|k-n|} \delta(k-k')$. What are $\Phi$ and $\lambda_k$?

b. This means the transformed image is uncorrelated across the rows and show that the pixels along each row can be modeled by the first order AR process of (11.84) with $a(k) = \rho$, $b(k) = 1$, and $\sigma^2(k) = (1 - \rho^2)\lambda_k$.

11.16 For images having separable covariance function with $\rho = 0.95$, find the optimum pairs $n, k(n)$ for DPCM transmission over a noisy channel with $p_e = 0.001$ employing the optimum mean square predictor. (Hint: $\beta^2 = (1 - \rho^2)^2$.)

11.17 Let the transition probabilities $q_0 = p(0|1)$ and $q_1 = p(1|0)$ be given. Assuming all the runs to be independent, their probabilities can be written as $g_i(l) = q_i(1 - q_i)^{l-1}$, $l \geq 1$, $i = 0$ (white), 1 (black)

a. Show that the average run lengths and entropies of white and black runs are $\mu_i = 1/q_i$, and $H_i = (1/q_i)[q_i \log_2 q_i + (1 - q_i) \log_2 (1 - q_i)]$. Hence the achievable compression ratio is $(H_0 P_0/\mu_0 + H_1 P_1/\mu_1)$, where $P_i = q_i/(q_0 + q_1)$, $i = 0, 1$ are the a priori probabilities of white and black pixels.

b. Suppose each run length is coded in blocks of $m$-bit words, each word representing the $M - 1$ run lengths in the interval $[kM, (k+1)M-1]$, $M = 2^n$, $k = 0, 1, \ldots$, and a block terminator code. Hence the average number of bits used for white and black runs will be $m \sum_{k=0}^{M-1} (k+1)P[kM \leq l \leq (k+1)M-1]$, $i = 0, 1$. What is the compression achieved? Show how to select $M$ to maximize it.
Section 11.1

Data compression has been a topic of immense interest in digital image processing. Several special issues and review papers have been devoted to this. For details and extended bibliographies:


Section 11.2

For entropy coding, Huffman coding, run-length coding, arithmetic coding, vector quantization, and related results of this section see papers in [3] and:


Section 11.3

For some early work on predictive coding, Delta modulation and DPCM see Oliver, Harrison, O’Neal, and others in *Bell Systems Technical Journal* issues of July 1952, May–June 1966, and December 1972. For more recent work:
For adaptive delta modulation algorithms and applications to image transmission, see [7], Cutler (pp. 898–906), Song, et al. (pp. 1033–1044) in [1b], Lei et al. in [1c]. For more recent work on DPCM of two-dimensional images, see Musmann in [4], Habibi (pp. 948–956) in [1b], Sharma et al. in [1c].


For adaptive DPCM, see Zschunk (pp. 1295–1302) and Habibi (pp. 1275–1284) in [1c], Jain and Wang [32], and:


### Section 11.4

For results related to the optimality of the KL transform, see Chapter 5 and the bibliography of that chapter. For the optimality of KL transform coding and bit allocations, see [6], [32], and:


### Section 11.5

For early work on transform coding and subsequent developments and examples of different transforms and algorithms, see Pratt and Andrews (pp. 515–554), Woods and Huang (pp. 555–573) in [2], and:


The concepts of fast KL transform and recursive block coding were introduced in [26 and Ref 17, Ch 5]. For details and extensions see [6], Meiri et al. (pp. 1728–1735) in [1e], Jain et al. in [8], and:


For results on two-source coding, adaptive transform coding, and the like, see Yan and Sakrison (pp. 1315–1322) in [1c], Jain and Wang [32], Tasto and Wintz (pp. 956–972) in [1b], Graham (pp. 336–346) in [1a], Chen and Smith in [1c], and:


**Section 11.6**

Hybrid coding principle, its analysis and relationship with semicausal models, and its applications can be found in [6, 8], Jones (Chapter 5) in [4], and:


Bibliography Chap. 11
Section 11.7

For interframe predictive coding, see [5, 6, 8], Haskell et al. (pp. 1339–1348) in [1c], Haskell (Chapter 6) in [4], and:


Interframe hybrid and transform coding techniques are discussed in [5, 6, 8, 9], Roese et al. (pp. 1329–1338), Natrajan and Ahmed (pp. 1323–1329) in [1c], and:


Section 11.8

For the optimal mean square encoding and decoding results and their application, we follow [6, 39] and:


For extended bibliography, see [6, 8].

Section 11.9

[1f] is devoted to coding of two tone images. Details of CCITT standards and various algorithms are available here. Some other useful references are Arps (pp. 222–276) in [4], Huang in [1c, 2], Musmann and Preuss in [1c], and:

Section 11.10

Color and multispectral coding techniques discussed here have been discussed by Limb et al. in [1c], Pratt in [1b], and:


Section 11.11

Several techniques not discussed in this chapter include nonuniform sampling techniques combined with interpolation (such as using splines), use of singular value decompositions, autoregressive (AR) model synthesis, and the like. Summary discussion and the relevant sources of these and other useful methods are given in [6, 8].